



THE UNIVERSITY OF  
CHICAGO



MPCS 51033 • AUTUMN 2017 • SESSION 10

---

# BACKENDS FOR MOBILE APPLICATIONS

# CLASS NEWS

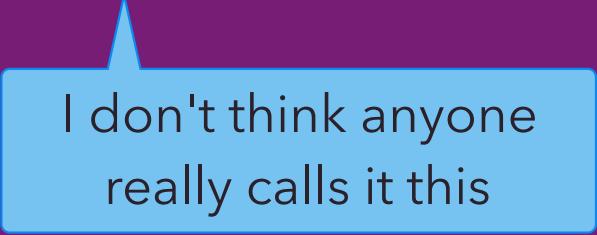
## CLASS NEWS

- No office hours Thursday
  - Email, slack, etc.
- Final Projects due December 7th at 11:59 pm.

# SERVERLESS ARCHITECTURES

# SERVERLESS ARCHITECTURES

- Serverless architectures can mean many things
  - BaaS
  - PaaS
- Function as a service (Faas) is the extreme realization of this architecture



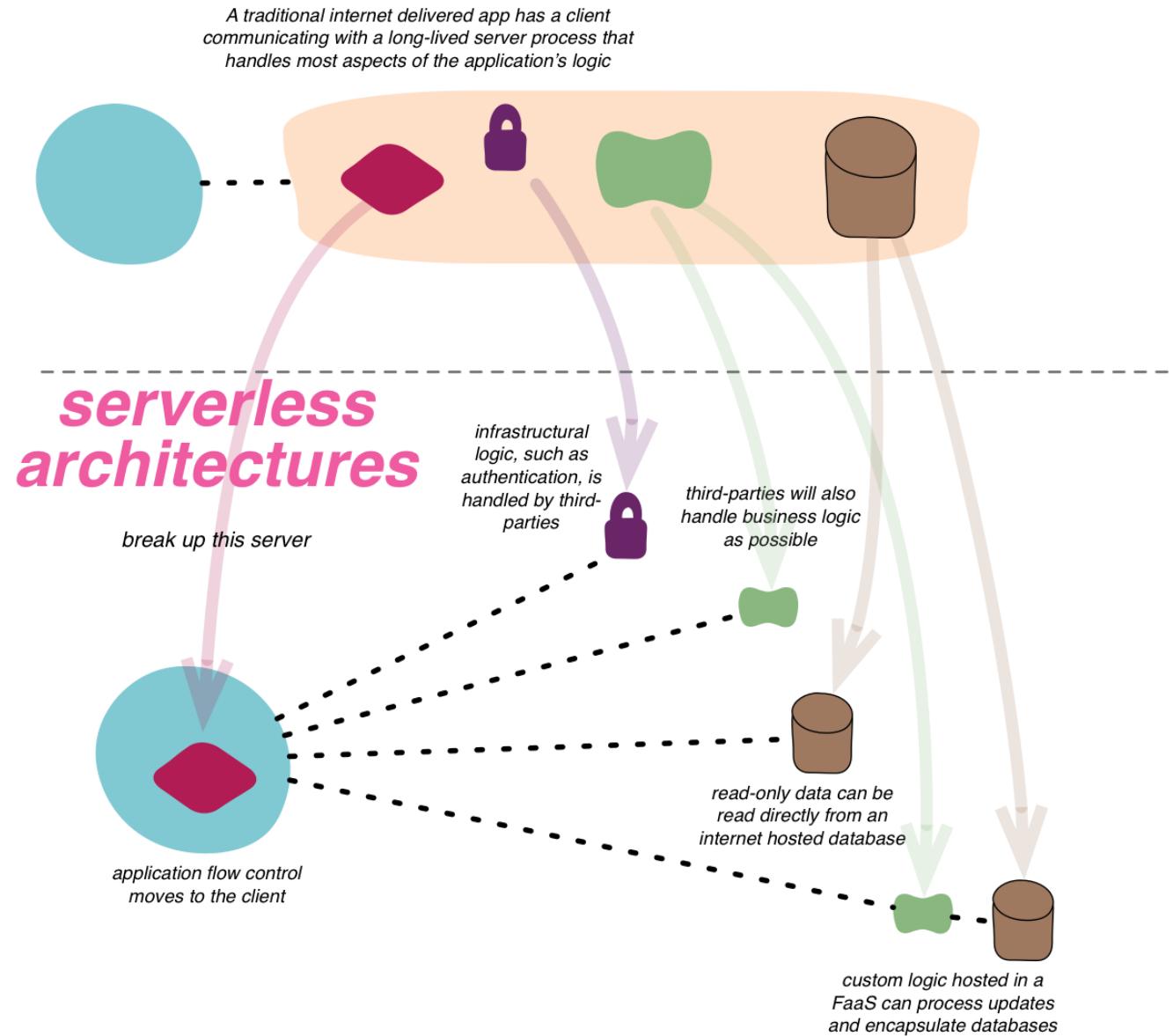
I don't think anyone  
really calls it this

# SERVERLESS ARCHITECTURES

- Despite what you may think, serverless architecture is really about moving as much behavior to the front end as possible
- Remove the need for traditional "always on" server

# SERVERLESS ARCHITECTURES

- Traditional vs. serverless architectures

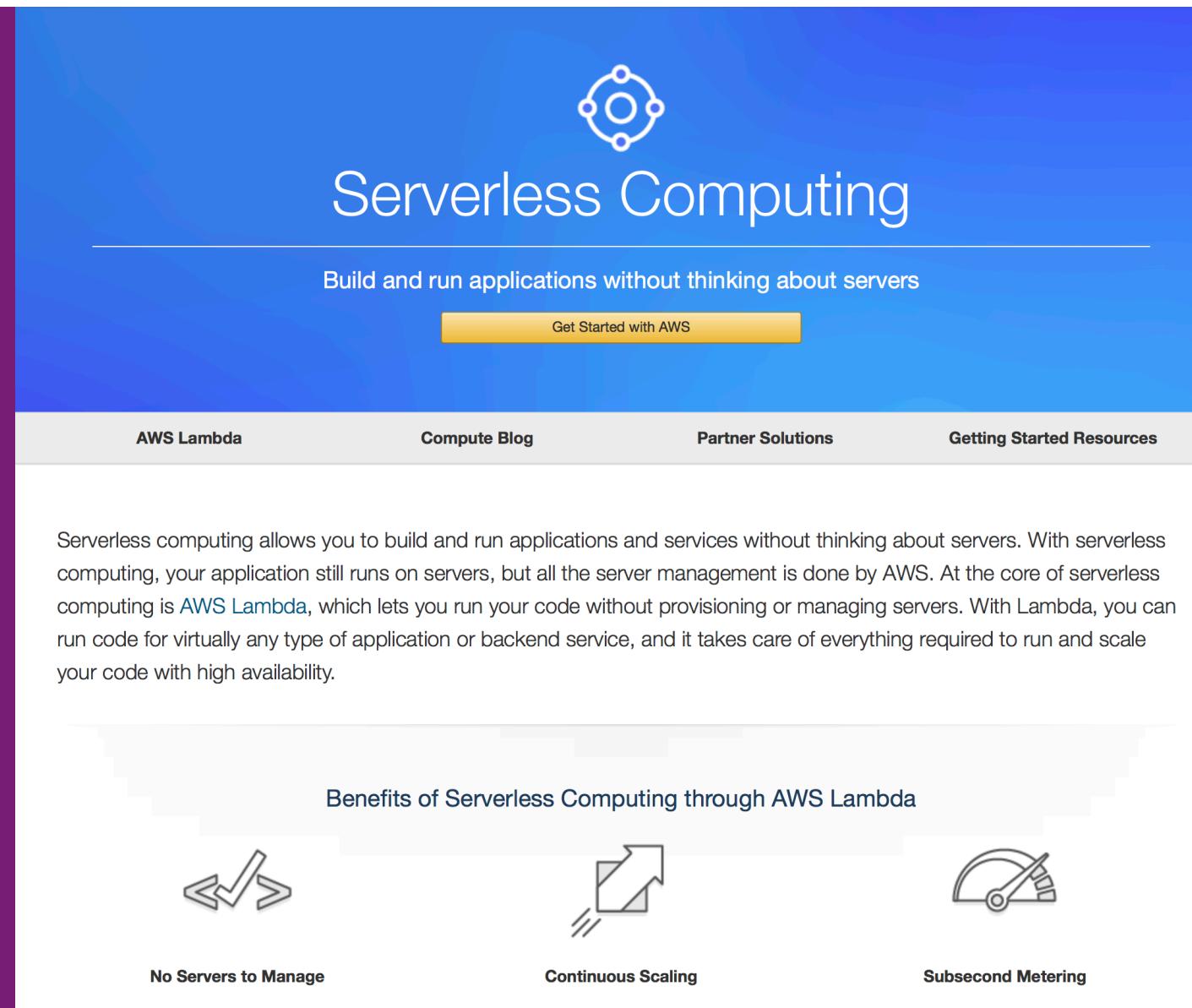


# SERVERLESS ARCHITECTURES

- Serverless applications have to rely on third-party services to accomplish tasks that are typically consolidated by a server
  - Each service might be provided by a different provider (eg. authentication with Google, database with Azure) or by a single provider (eg. Firebase)
  - The fact that some services are by the same provider is more of an administrative convenience...serverless is about abstraction

# SERVERLESS ARCHITECTURES

- Key players
  - AWS Lambda
  - Microsoft
  - Google Cloud Functions
  - Firebase Cloud Functions



The screenshot shows the AWS Serverless Computing landing page. At the top center is a circular icon with three dots and lines. Below it is the title "Serverless Computing". A subtitle reads "Build and run applications without thinking about servers". A yellow "Get Started with AWS" button is visible. The page features a navigation bar with links for "AWS Lambda", "Compute Blog", "Partner Solutions", and "Getting Started Resources". The main content area contains text explaining what serverless computing is and how AWS Lambda fits into it. It highlights benefits such as "No Servers to Manage", "Continuous Scaling", and "Subsecond Metering".

Serverless computing allows you to build and run applications and services without thinking about servers. With serverless computing, your application still runs on servers, but all the server management is done by AWS. At the core of serverless computing is [AWS Lambda](#), which lets you run your code without provisioning or managing servers. With Lambda, you can run code for virtually any type of application or backend service, and it takes care of everything required to run and scale your code with high availability.

Benefits of Serverless Computing through AWS Lambda

- No Servers to Manage
- Continuous Scaling
- Subsecond Metering

# SERVERLESS ARCHITECTURES

- Serverless logic tied to services
  - Parse (RIP)
  - Realm
  - Firebase

The screenshot shows a blog post titled "Serverless Logic with Realm: Introducing Realm Functions". The post is by the "Realm Team" and was published on "May 23 2017". It features a snippet of JavaScript code demonstrating how to use the Wit API with Realm. Below the post, there's a summary of the announcement: "Today we're announcing Realm Functions, a new part of Realm that makes building server-side functionality a lot easier for mobile developers. Now, you can make server-side features without enlisting backend developers, plus you get all the benefits of building on top of the Realm Mobile Platform: you don't need to add another endpoint to a server, and then write the serialization and networking code that would let you connect with it. You just connect your app to Realm, write a Realm Function in your web dashboard, and watch your code execute reactively as data streams in. Today's release is a beta, and it's available today to everyone, whether you're building an app in an enterprise-scale team or for a small side project." Navigation links include "Go to News", "Transcript", and "About the Speaker". Social sharing icons for Twitter, Facebook, and Email are also present.

Realm Tasks With Wit

Regex

```
var Wit = require("node-wit").Wit;
var WIT_ACCESS_TOKEN = "YBXCZ2LQeJ...";
var witClient = new Wit({accessToken: WIT_ACCESS_TOKEN});

module.exports = function(change_event) {
  var realm = change_event.realm;
  var changes = change_event.changes.Task;
  var taskIndexes = changes.modifications;
  console.log("Change detected: " + changes);
}
```

Realm Team

May 23 2017

Go to News

Transcript

About the Speaker

Twitter

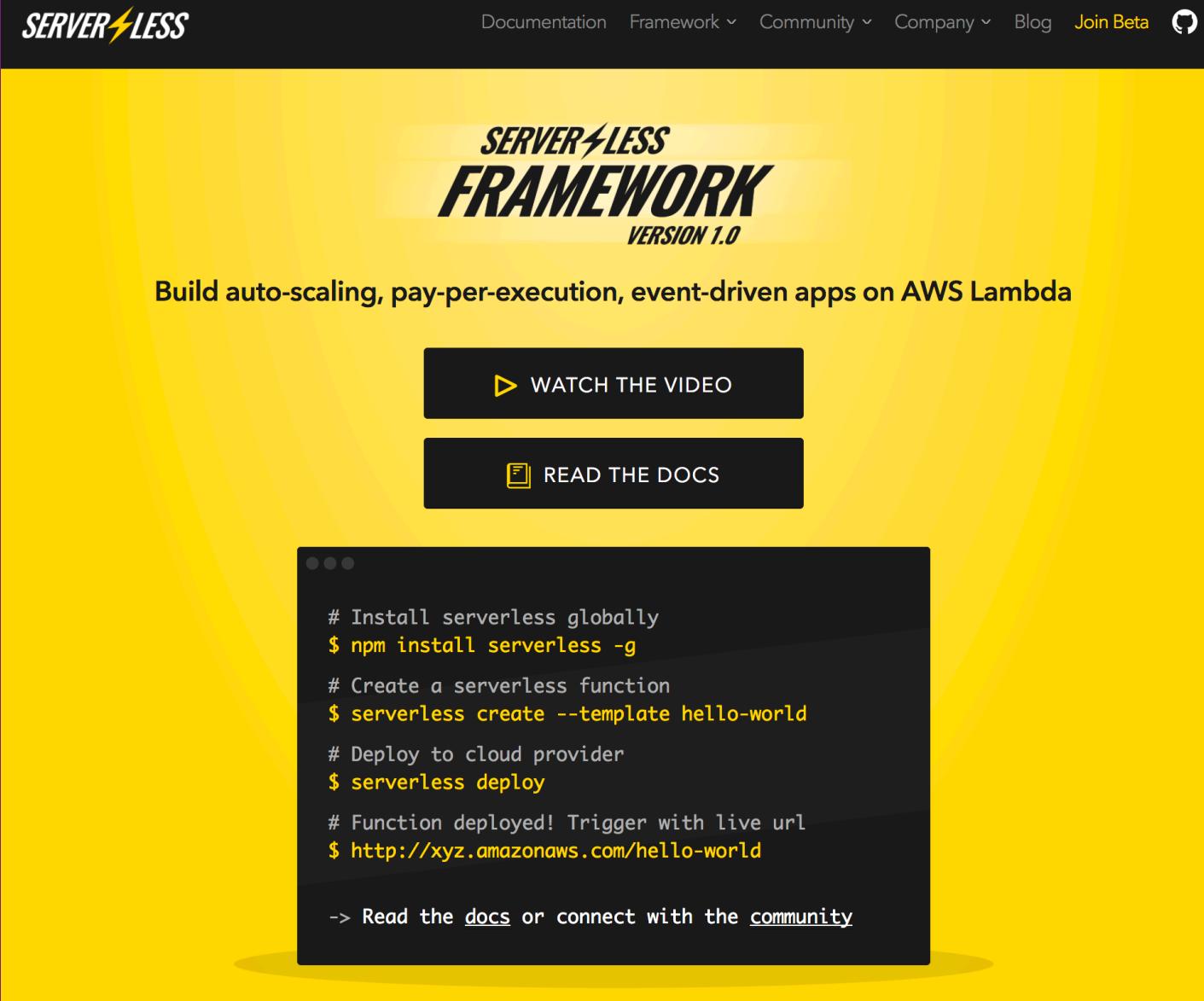
Facebook

Email

Today we're announcing Realm Functions, a new part of Realm that makes building server-side functionality a lot easier for mobile developers. Now, you can make server-side features without enlisting backend developers, plus you get all the benefits of building on top of the Realm Mobile Platform: you don't need to add another endpoint to a server, and then write the serialization and networking code that would let you connect with it. You just connect your app to Realm, write a Realm Function in your web dashboard, and watch your code execute reactively as data streams in. Today's release is a beta, and it's available today to everyone, whether you're building an app in an enterprise-scale team or for a small side project.

# SERVERLESS ARCHITECTURES

- Frameworks for server less
  - Write once, deploy everywhere



The screenshot shows the official Serverless Framework website. At the top, there's a navigation bar with links for Documentation, Framework, Community, Company, Blog, Join Beta, and a GitHub icon. The main title "SERVERLESS FRAMEWORK VERSION 1.0" is prominently displayed in the center, with "VERSION 1.0" in smaller text below it. Below the title, a subtitle reads "Build auto-scaling, pay-per-execution, event-driven apps on AWS Lambda". There are two dark call-to-action buttons: "WATCH THE VIDEO" with a play icon and "READ THE DOCS" with a document icon. A large code block in the bottom right corner shows a terminal session with commands to install the framework globally, create a function, and deploy it to AWS Lambda, along with a live URL. A note at the bottom of the code block encourages users to "Read the [docs](#) or connect with the [community](#)".

Documentation Framework Community Company Blog Join Beta

**SERVERLESS FRAMEWORK**  
VERSION 1.0

Build auto-scaling, pay-per-execution, event-driven apps on AWS Lambda

► WATCH THE VIDEO

READ THE DOCS

```
# Install serverless globally
$ npm install serverless -g
# Create a serverless function
$ serverless create --template hello-world
# Deploy to cloud provider
$ serverless deploy
# Function deployed! Trigger with live url
$ http://xyz.amazonaws.com/hello-world

-> Read the docs or connect with the community
```

# SERVERLESS ARCHITECTURES

- A major consideration
  - The runtime of server less functions can be extremely limited
  - Server only active during process (pay per time)
  - Data can be returned in the response
  - Data saved requires an outgoing request



All about  
tradeoffs

# SERVERLESS ARCHITECTURES

- Benefits
  - Costs (pay for what you use)
  - Reduced complexity (of running a server)
  - Scaling

# SERVERLESS ARCHITECTURES

- Costs
  - Conceptual
  - Third-party dependency
  - Control (whims of big companies)
  - Local development and testing

# AWS LAMBDA

# AWS LAMBDA

## Compute

- EC2
- EC2 Container Service
- Lightsail 
- Elastic Beanstalk
- Lambda
- Batch

## Developer Tools

- CodeStar
- CodeCommit
- CodeBuild
- CodeDeploy
- CodePipeline
- X-Ray

## Analytics

- Athena
- EMR
- CloudSearch
- Elasticsearch Service
- Kinesis
- Data Pipeline
- QuickSight 

## Application Services

- Step Functions
- SWF
- API Gateway
- Elastic Transcoder

## Messaging

- Simple Queue Service
- Simple Notification Service
- SES

AWS has a lot  
of offerings

## Storage

- S3
- EFS
- Glacier
- Storage Gateway

## Management Tools

- CloudWatch
- CloudFormation
- CloudTrail
- Config
- OpsWorks
- Service Catalog
- Trusted Advisor
- Managed Services

## Artificial Intelligence

- Lex
- Polly
- Rekognition
- Machine Learning

## Database

- RDS
- DynamoDB
- Amazon Aurora

## Internet Of Things

- AWS IoT

## Business Productivity

- WorkDocs
- WorkMail
- Amazon Chime 

## Desktop & App Streaming

# AWS LAMBDA

- AWS Lambda  
serverless  
architecture

The screenshot shows the AWS Lambda service landing page. At the top, there's a navigation bar with the AWS logo, 'Services' dropdown, 'Resource Groups' dropdown, a bell icon, and user information ('T. Andrew Binkowski'). Below the navigation is a sidebar titled 'AWS Lambda' with 'Dashboard' and 'Functions' options. The main content area has a dark header 'COMPUTE' and a large title 'AWS Lambda' followed by a subtitle: 'lets you run code without thinking about servers.' A paragraph explains the cost model: '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.' To the right, there's a 'Get started' section with a 'Create a function' button, and a 'More resources' sidebar with links to Documentation, API reference, Serverless Application Model, SAM Local, and Forums.

**AWS Lambda**

lets you run code without thinking about 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.

## How it works

Upload your code to AWS Lambda, set it up to trigger from other AWS services, HTTP endpoints, or in-app activity, and let Lambda run and scale your code with high availability — all without provisioning or managing any servers.

[Read more in FAQs ↗](#)

## Related services

**Get started**

Author a Lambda function from scratch, or choose from one of our preconfigured examples.

**Create a function**

**More resources**

[Documentation](#)

[API reference](#)

[Serverless Application Model](#)

[SAM Local](#)

[Forums](#)

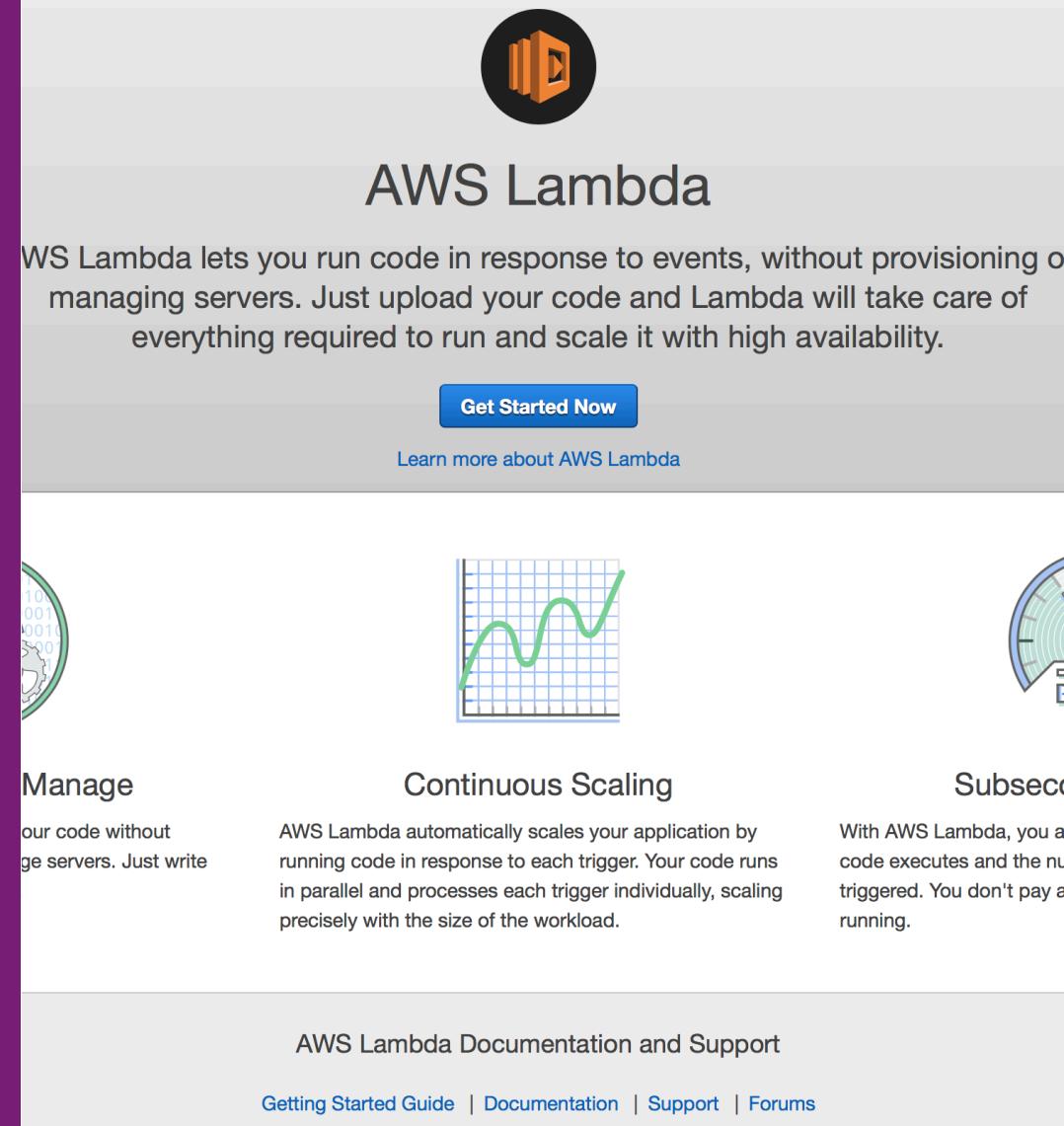
# AWS LAMBDA

- The Lambda free tier includes 1M free requests per month and 400,000 GB-seconds of compute time per month
- The memory size you choose for your Lambda functions determines how long they can run in the free tier
- The Lambda free tier does not automatically expire at the end of your 12 month AWS Free Tier term, but is available forever

Memory (MB)	Free tier seconds per month	Price per 100ms (\$)
128	3,200,000	0.000000208
192	2,133,333	0.000000313
256	1,600,000	0.000000417
320	1,280,000	0.000000521
384	1,066,667	0.000000625
448	914,286	0.000000729
512	800,000	0.000000834
576	711,111	0.000000938
640	640,000	0.000001042
704	581,818	0.000001146
768	533,333	0.000001250
832	492,308	0.000001354
896	457,143	0.000001459
960	426,667	0.000001563
1024	400,000	0.000001667
1088	376,471	0.000001771
1152	355,556	0.000001875
1216	336,842	0.000001980

# AWS LAMBDA

- Create an AWS account
- Create admin account (optional)
- Download AWS CLI
  - Homebrew or pip
- Create and deploy



The screenshot shows the AWS Lambda landing page. At the top right is the Lambda logo (an orange 3D block icon). Below it is the heading "AWS Lambda". A main text block explains that Lambda lets you run code in response to events without provisioning or managing servers, with a "Get Started Now" button and a "Learn more about AWS Lambda" link. To the left, there's a "Manage" section with a gear icon and a brief description of running code without managing servers. To the right, there's a "Continuous Scaling" section with a graph icon and a description of automatically scaling applications. At the bottom, there's a "Subsec..." section and links for "AWS Lambda Documentation and Support", "Getting Started Guide", "Documentation", "Support", and "Forums".

AWS Lambda

WS Lambda lets you run code in response to events, without provisioning or managing servers. Just upload your code and Lambda will take care of everything required to run and scale it with high availability.

Get Started Now

Learn more about AWS Lambda

Manage

our code without managing servers. Just write

Continuous Scaling

AWS Lambda automatically scales your application by running code in response to each trigger. Your code runs in parallel and processes each trigger individually, scaling precisely with the size of the workload.

Subsec...

AWS Lambda Documentation and Support

Getting Started Guide | Documentation | Support | Forums

# AWS LAMBDA

- Blueprints are sample configurations of event sources and Lambda functions
- Choose a blueprint that best aligns with your desired scenario and customize

Lambda > New function

**Select blueprint**

Configure triggers  
Configure function  
Review

**Select blueprint**

Blueprints are sample configurations of event sources and Lambda functions. Choose a blueprint that best aligns with your desired scenario and customize as needed, or skip this step if you want to author a Lambda function and configure an event source separately. Except where otherwise noted, blueprints are licensed under [CC0](#).

Welcome to AWS Lambda! You can get started on creating your first Lambda function by choosing one of the blueprints below.

Select runtime ▾ Filter << < Viewing 1-9 of 94 > >

Blank Function	kinesis-firehose-syslog-to-json	alexa-skill-kit-sdk-factskill
Configure your function from scratch. Define the trigger and deploy your code by stepping through our wizard. <a href="#">custom</a>	An Amazon Kinesis Firehose stream processor that converts input records from RFC3164 Syslog format to JSON. <a href="#">nodejs · kinesis-firehose</a>	Demonstrate a basic fact skill built with the ASK NodeJS SDK <a href="#">nodejs6.10 · alexa</a>
batch-get-job-python27	kinesis-firehose-apachelog-to...	cloudfront-modify-response-h...
Returns the current status of an AWS Batch Job. <a href="#">python2.7 · batch</a>	An Amazon Kinesis Firehose stream processor that converts input records from Apache Common Log format to <a href="#">python2.7 · kinesis-firehose</a>	Blueprint for modifying CloudFront response header implemented in NodeJS. <a href="#">nodejs · cloudfront · response header</a>

## AWS LAMBDA

- Warning: the configurations for the blueprints don't all work out of the box

### hello-world-python

A starter AWS Lambda function.

---

python2.7



# AWS LAMBDA

- Create functions
  - In-browser editor
  - External gzip and push

Lambda > Functions > Create function > Using blueprint hello-world-python

## Basic information [Info](#)

Name\*  
hello-world-2017-autym

Role\*  
Defines the permissions of your function. Note that new roles may not be available for a few minutes after creation. [Learn more](#) about Lambda execution roles.

Create new role from template(s) ▾

Lambda will automatically create a role with permissions from the selected policy templates. Note that basic Lambda permissions (logging to CloudWatch) will automatically be added. If your function accesses a VPC, the required permissions will also be added.

Role name\*  
Enter a name for your new role.  
Admin

Policy templates  
Choose one or more policy templates. A role will be generated for you before your function is created. [Learn more](#) about the permissions that each policy template will add to your role.

Basic Edge Lambda permissions X

# AWS LAMBDA

- Create functions
  - Set roles
  - Set permissions

Lambda > Functions > Create function > Using blueprint hello-world-python

## Basic information [Info](#)

Name\*

Role\*  
Defines the permissions of your function. Note that new roles may not be available for a few minutes after creation. [Learn more](#) about Lambda execution roles.

Lambda will automatically create a role with permissions from the selected policy templates. Note that basic Lambda permissions (logging to CloudWatch) will automatically be added. If your function accesses a VPC, the required permissions will also be added.

Role name\*  
Enter a name for your new role.

Policy templates  
Choose one or more policy templates. A role will be generated for you before your function is created. [Learn more](#) about the permissions that each policy template will add to your role.

# AWS LAMBDA

- Functions can be written in JS, Python, Java, and Go

Select blueprint

Configure triggers

Configure function

Review

## Configure function

A Lambda function consists of the custom code you want to execute. [Learn more](#) about Lambda functions.

Name\*

Description

Runtime\*

### Lambda function code

Provide the code for your function. Use the editor if your code does not require custom libraries (other than boto3). If you need custom libraries, you can upload your code and libraries as a .ZIP file.

Code entry type

```
1 from __future__ import print_function
2
3 import json
4
5 print('Loading function')
6
7
8 def lambda_handler(event, context):
9     #print("Received event: " + json.dumps(event, indent=2))
10    print("value1 = " + event['key1'])
11    print("value2 = " + event['key2'])
12    print("value3 = " + event['key3'])
```

Limited third party modules

# AWS LAMBDA

- Triggers define when your function runs
  - Event based on other AWS Services (push to S3; filter photo)
  - HTTP API
  - Mobile API
  - Scheduled Events (cron)

Functions can be triggered by different events

# AWS LAMBDA

- Function handler is the main function called
  - `lambda_function.lambda_handler` is the default console function
  - Name doesn't matter

## Lambda function code

Provide the code for your function. Use the editor if your code does not require custom libraries (other libraries, you can upload your code and libraries as a .ZIP file.

Code entry type

Edit code inline

```
1 from __future__ import print_function
2
3 import json
4
5 print('Loading function')
6
7
8 def lambda_handler(event, context):
9     #print("Received event: " + json.dumps(event, indent=2))
10    print("value1 = " + event['key1'])
11    print("value2 = " + event['key2'])
12    print("value3 = " + event['key3'])
13    return event['key1'] # Echo back the first key value
14    #raise Exception('Something went wrong')
```

main function

You can define Environment Variables as key-value pairs that are accessible from your function code. To change environment variables, edit the `environment_variables` section of the Lambda function configuration. You can also set environment variables directly in the Lambda function configuration without changing the code. Learn more. For storing sensitive information, we recommend using AWS Secrets Manager or the console's encryption helpers.

# AWS LAMBDA

- Parameters
  - Event - data passed in
  - Context - runtime information

## Lambda function code

Provide the code for your function. Use the editor if your code does not require custom libraries (other libraries, you can upload your code and libraries as a .ZIP file.

Code entry type

Edit code inline

```
1 from __future__ import print_function
2
3 import json
4
5 print('Loading function')
6
7
8 def lambda_handler(event, context):
9     #print("Received event: " + json.dumps(event, indent=2))
10    print("value1 = " + event['key1'])
11    print("value2 = " + event['key2'])
12    print("value3 = " + event['key3'])
13    return event['key1'] # Echo back the first key value
14    #raise Exception('Something went wrong')
15
```

You can define Environment Variables as key-value pairs that are accessible from your function code. To change environment variables, edit them in the Lambda function configuration. You can also change environment settings without the need to change function code. [Learn more](#). For storing sensitive information, we recommend using AWS Secrets Manager or Lambda environment variables.

# AWS LAMBDA

- Return values depends on how you set up the function
- Functions need to return properly to test

```
unction.py
from __future__ import print_function

import json

print('Loading function')


def lambda_handler(event, context):
    return {"statusCode": 200, \
            "headers": {"Content-type": "application/json"}, \
            "body": "{\"count\": 5, \"message\": \"hello\"}", \
            "isBase64Encoded": "false"}
```

# AWS LAMBDA

## ▼ Advanced settings

These settings allow you to control the code execution performance and costs for your Lambda function. Changing your resource settings (by selecting memory) or changing the timeout may impact your function cost. [Learn more](#) about how Lambda pricing works.

Memory (MB)\*  ⓘ

Timeout\*  min  sec

Sufficient for Hello  
World

AWS Lambda will automatically retry failed executions for asynchronous invocations. You can additionally optionally configure Lambda to forward payloads that were not processed to a dead-letter queue (DLQ), such as an SQS queue or an SNS topic. Learn more about Lambda's [retry policy](#) and [DLQs](#). **Please ensure your role has appropriate permissions to access the DLQ resource.**

DLQ Resource  ⓘ

All AWS Lambda functions run securely inside a default system-managed VPC. However, you can optionally configure Lambda to access resources, such as databases, within your custom VPC. [Learn more](#) about accessing VPCs within Lambda. **Please ensure your role has**

# AWS LAMBDA

Lambda > Functions > hello-world-2017-autym

ARN - arn:aws:lambda:us-east-2:735044932848:function:hello-world-2017-autym

## hello-world-2017-autym

Qualifiers ▾

Actions ▾

test1 ▾

Test

Execution result: succeeded ([logs](#))

▼ Details

The area below shows the result returned by your function execution.

"value1"

Test

### Summary

Code SHA-256

6chaTsf/AFve6WIJ3HrQFLvl  
t7xggyqTeTa8DvnxQeA=

Request ID

# AWS LAMBDA

The screenshot shows the AWS Lambda function details page for 'hello-world-2017-autym'. The top navigation bar shows 'Lambda > Functions > hello-world-2017-autym'. To the right, the ARN is listed as 'arn:aws:lambda:us-east-2:735044932848:f...'. Below the navigation, the function name 'hello-world-2017-autym' is displayed. On the right side, there are three buttons: 'Qualifiers ▾', 'Actions ▾', and a dropdown menu set to 'test1'. The 'Actions' menu is open, showing options: 'Publish new version', 'Create alias', 'Delete function', and 'Export function'. A green checkmark icon next to 'Execution result: succeeded (logs)' indicates a successful deployment. A 'Details' section below it contains the message: 'The area below shows the result returned by your function execution.'

Lambda > Functions > hello-world-2017-autym

ARN - arn:aws:lambda:us-east-2:735044932848:f...

hello-world-2017-autym

Qualifiers ▾

Actions ▾

test1

Publish new version

Create alias

Delete function

Export function

Execution result: succeeded ([logs](#))

▼ Details

The area below shows the result returned by your function execution.

- Review your Lambda function before deploying
- You can test and edit

# AWS LAMBDA

- Test function
  - Response
  - Log output
  - Runtime summary

Execution result: succeeded ([logs](#))

▼ Details

The area below shows the result returned by your function execution.

```
"value1"
```

Summary

Code SHA-256  
6chaTsf/AFve6WIJ3HrQFLvl  
t7xgyqTeTa8DvnxQeA=

Request ID  
9cd2f27a-d3de-11e7-ad1b-  
97d23359cdfc

Duration  
0.26 ms

Billed duration  
100 ms

Resources configured

128 MB

Max memory used  
19 MB



Test

# AWS LAMBDA

Print statements go  
to logging

## Log output

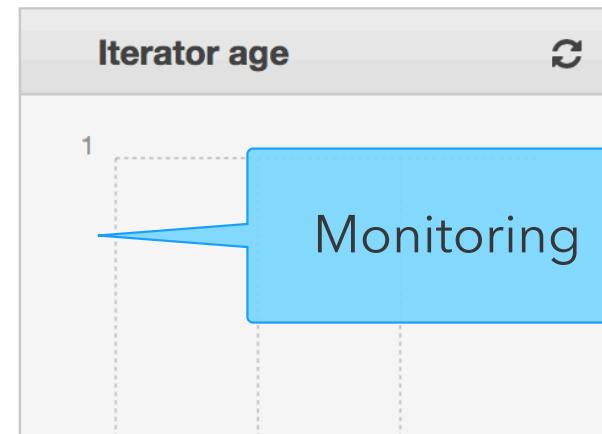
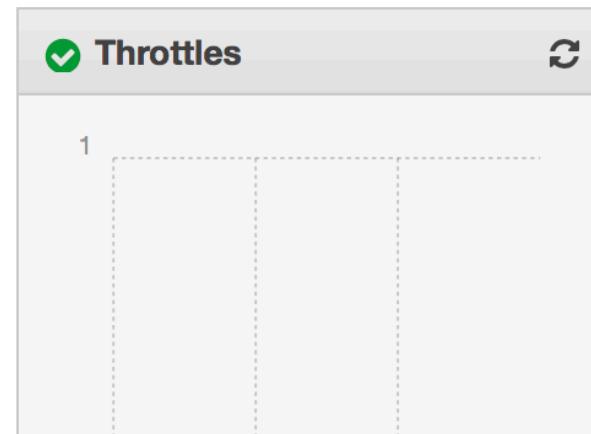
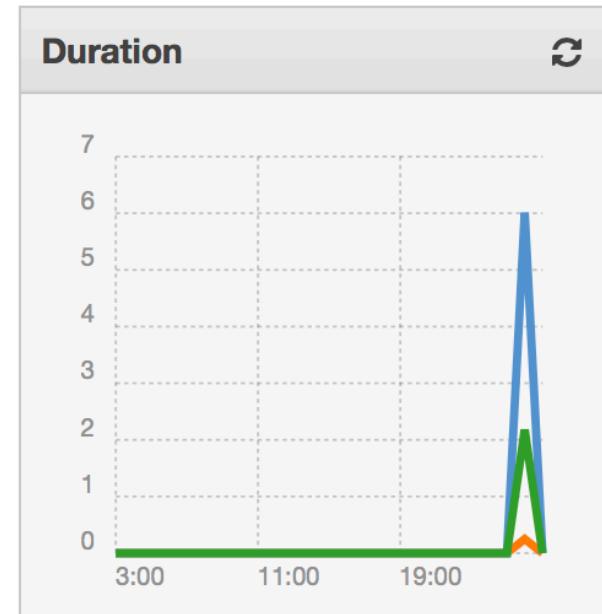
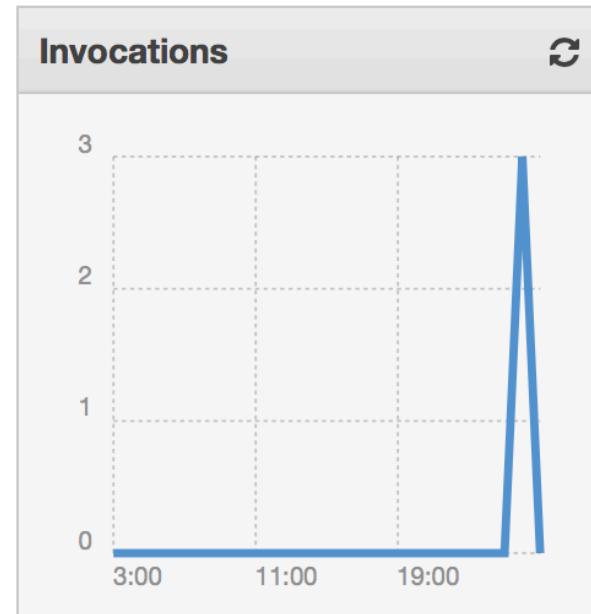
The area below shows the logging calls in your code. These correspond to a single row within the CloudWatch log group corresponding to this Lambda function. [Click here](#) to view the CloudWatch log group.

```
START RequestId: 9cd2f27a-d3de-11e7-ad1b-97d23359cdfc Version: $LATEST
value1 = value1
value2 = value2
value3 = value3
END RequestId: 9cd2f27a-d3de-11e7-ad1b-97d23359cdfc
REPORT RequestId: 9cd2f27a-d3de-11e7-ad1b-97d23359cdfc Duration: 0.26 ms          Billed Duration: 100 ms      Memory Size: 128 MB Max Memory Used: 19 MB
```

# AWS LAMBDA

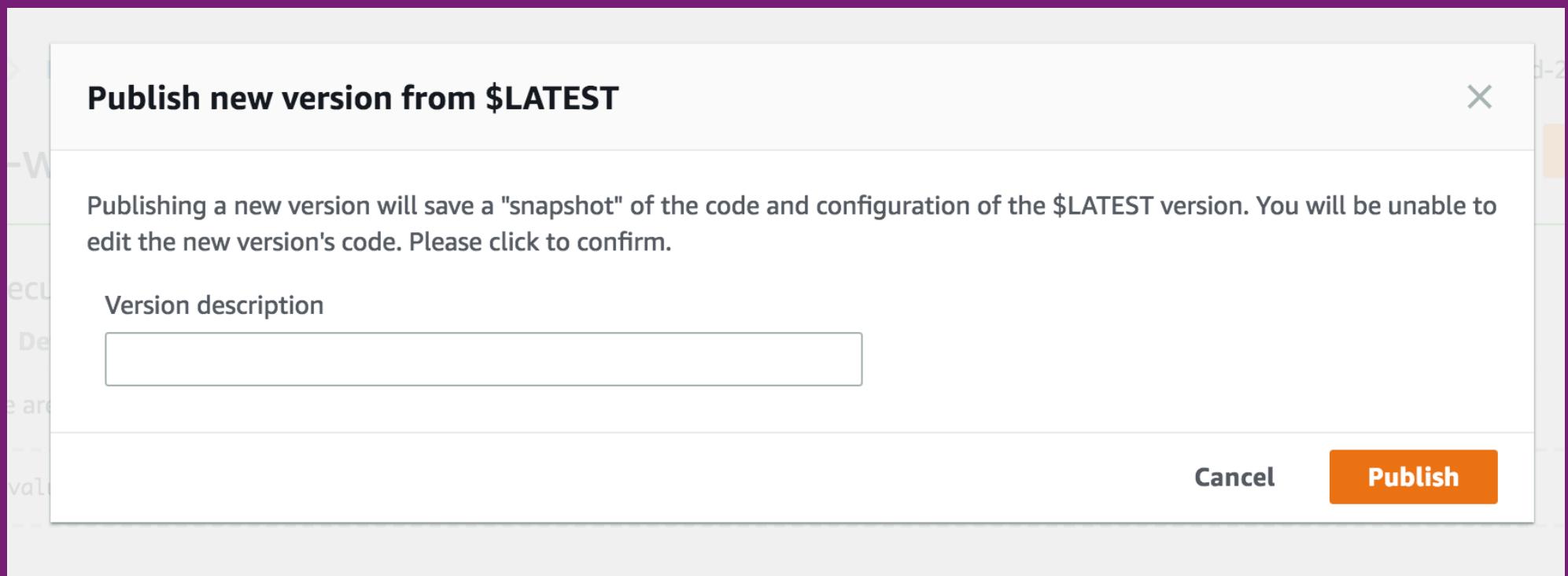
- Monitor your function

CloudWatch metrics at a glance (last 24 hours)



Monitoring

# AWS LAMBDA



- Publish

# AWS LAMBDA

- Set up a trigger

### Add trigger

Configure your Lambda function **helloWorld** to respond to events from the selected trigger. Click on the box below to select your trigger type.



Lambda

Filter integrations

- API Gateway
- AWS IoT
- CloudFront
- CloudWatch Events - Schedule
- CloudWatch Logs
- CodeCommit
- Cognito Sync Trigger
- DynamoDB

Cancel    Submit

# AWS LAMBDA

- More security hoops to jump through
- In my experience, you may need to delete the trigger after updating

Add trigger

Configure your Lambda function **helloWorld** to respond to events from the selected trigger. Click on the box below to select your trigger type.

API Gateway  → Lambda 

Please go to the [IAM console](#) to configure the security for your API endpoint. ✖

We'll set up an API Gateway endpoint with a [proxy integration type](#) (learn more about the [input](#) and [output](#) format for your function). Any method (GET, POST, etc.) will trigger your Lambda function. To set up more advanced method mappings or subpath routes, visit [Amazon API Gateway console](#).

API name: LambdaMicroservice  ⓘ

Deployment stage: prod  ⓘ

Security: AWS IAM  ⓘ

Lambda will add the necessary permissions for Amazon API Gateway to invoke your Lambda function. [Learn more](#) about the Lambda permissions model.

[Cancel](#) [Submit](#)

# AWS LAMBDA

hello-world-2017-autym:1

Version: 1 Actions test1 Test

Execution result: succeeded (logs)  
► Details

Successfully added the trigger nt1vlpjhxf to function hello-world-2017-autym. The function is now receiving events from the trigger. X

Configuration Triggers Monitoring Go to API

API Gateway: LambdaMicroservice Delete  
arn:aws:execute-api:us-east-2:735044932848:nt1vlpjhxf/prod/ANY/hello-world-2017-autym  
► Method: ANY Resource path: /hello-world-2017-autym Authorization: AWS\_IAM

# AWS LAMBDA

The screenshot shows the AWS API Gateway test interface. On the left, the sidebar includes sections for APIs (LambdaMicroservice), Resources (Resources, Stages, Authorizers, Models, Documentation, Binary Support, Dashboard), Usage Plans, API Keys, Custom Domain Names, Client Certificates, and Settings. The main area shows a test configuration for the '/helloWorld' resource with the 'ANY' method selected. The configuration includes:

- Query Strings:** A text input field containing '{helloWorld}'.
- Headers:** A text input field containing '{helloWorld}'. Below it is a note: "Use a colon (:) to separate header name and value, and new lines to declare multiple headers. eg. Accept:application/json."
- Stage Variables:** A note stating "No stage variables exist for this method."
- Client Certificate:** A note stating "No client certificates have been generated."
- Request Body:** A text input field with the number '1' at the top.

On the right, the results section displays:

- Response Headers:** An empty list: {}.
- Logs:** A detailed execution log for the request. The log starts with "Execution log for request test-request" and continues with a timestamped sequence of API Gateway and Lambda logs, including X-Amz-Date, Authorization, and various AWS service identifiers.

- Test your API (note that to be successful, need to return proper code)

# AWS LAMBDA

```
import json

print('Loading function')

def lambda_handler(event, context):
    return {"statusCode": 200, \
            "headers": {"Content-type": "application/json"}, \
            "body": "{\"count\": 5, \"message\": \"hello\"}", \
            "isBase64Encoded": "false"}
```

- Test your API (note that to be successful, need to return proper code)

# AWS LAMBDA

- CloudWatch trigger
- Schedule
- AWS Event



Rule

Pick an existing rule, or create a new one.

**Create a new rule** ▾

Select or create a new rule

Rule name\*

Enter a name to uniquely identify your rule.

Rule description

Provide an optional description for your rule.

Rule type

Trigger your target based on an event pattern, or based on an automated schedule.

Event pattern

Schedule expression

Schedule expression\*

Self-trigger your target on an automated schedule using Cron or rate expressions. Cron expressions are in UTC.

e.g. rate(1 day), cron(0 17 ? \* MON-FRI \*)

# AWS LAMBDA

- Limits to consider for lambda

## AWS Lambda Resource Limits

Resource	Default Limit
Ephemeral disk capacity ("tmp" space)	512 MB
Number of file descriptors	1,024
Number of processes and threads (combined total)	1,024
Maximum execution duration per request	300 seconds
Invoke request body payload size (RequestResponse)	6 MB
Invoke request body payload size (Event)	128 K
Invoke response body payload size (RequestResponse)	6 MB

The following table lists the Lambda account limits per region.

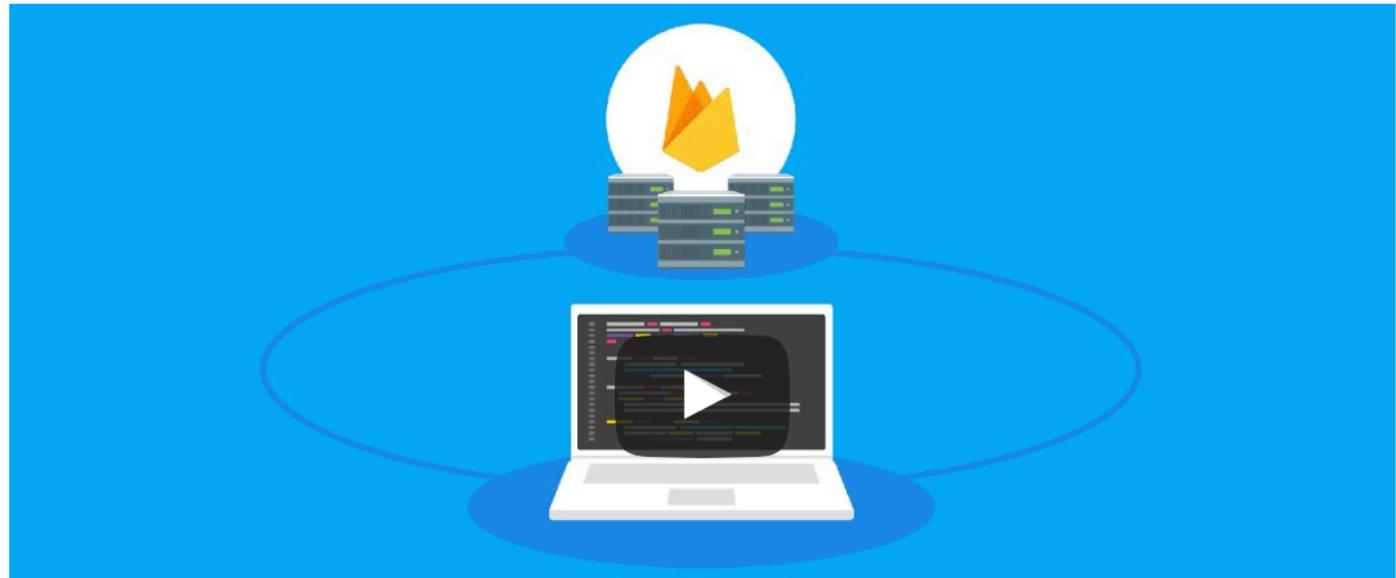
## AWS Lambda Account Limits Per Region

Resource	Default Limit
Concurrent executions (see <a href="#">Lambda Function Concurrent Executions</a> ).	1000

# FIREBASE CLOUD FUNCTIONS VS LAMBDA

# 🔥 FIREBASE CLOUD FUNCTIONS

- <https://youtu.be/vr0Gfvp5v1A>



Cloud Functions  
for Firebase



# CLOUD FUNCTIONS

- Integrates with Firebase in different ways
  - Realtime Database Triggers
  - Firebase Authentication Triggers
  - Firebase Analytics Triggers
  - Cloud Storage Triggers
  - Cloud Pub/Sub Triggers
  - HTTP Triggers





## FIREBASE

## CLOUD FUNCTIONS

- Overall the functionality and offerings are comparable
  - Deploy functions in the cloud
  - Integrations with other offering (aws and Google ecosystem)
- Devil is in the details

# CREATING FUNCTIONS

## CREATING FUNCTIONS

- Lambda
  - Nice testing environment
  - In-browser editor
  - Language support
  - More trigger variety (cron)
  - Ton of boilerplate code



# AWS Lambda

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

[Get started with AWS Lambda](#)

# CREATING FUNCTIONS

- Firebase
  - Limited (now) to JS
  - Functions can be tied to other Firebase/Google events
  - Fewer triggers

The functions you write can respond to events generated by these other Firebase and Google Cloud features:

- [Realtime Database Triggers](#)
- [Firebase Authentication Triggers](#)
- [Google Analytics for Firebase Triggers](#)
- [Cloud Storage Triggers](#)
- [Cloud Pub/Sub Triggers](#)
- [HTTP Triggers](#)

# DEPLOYING

# DEPLOYING

- Deployment and setup is similar
- Lamda does have the in-brower development and deployment

`aws deploy`

`firebase deploy`

# TESTING

# TESTING

- Lambda
  - Great testing environment
  - Test functions and API

Lambda > Functions > helloWorld

Qualifiers ▾ Test Actions ▾

Code Configuration Triggers Tags Monitor

Code entry type Edit code inline

```
1 from __future__ import print_function
2
3 import json
4
5 print('Loading function')
6
7
8 def lambda_handler(event, context):
9     #print("Received event: " + json.dumps(event, indent=2))
10    print("value1 = " + event['key1'])
11    print("value2 = " + event['key2'])
12    print("value3 = " + event['key3'])
13
14    return {"statusCode": 200, \
15            "headers": {"Content-Type": "application/json"}, \
16            "body": {"message": "hello world"}}
17
18    #return event['key1'] # Echo back the key provided in the function
19    #raise Exception('Something went wrong')
20
```

# TESTING

- Lambda
  - Debug console and logging provides useful information

The screenshot shows the AWS Lambda Test interface. At the top, there are three tabs: Qualifiers, Test (which is selected), and Actions. Below the tabs, a message indicates "Execution result: succeeded (logs)". A note below states: "The area below shows the result returned by your function execution. [Learn more](#) about returning results from your function." The results section displays a JSON object:

```
{  
  "statusCode": 400,  
  "headers": {  
    "Content-Type": "application/json"  
  },  
  "body": "{\"error\":\"Email or phone in use\""}  
}
```

Below the results, there are two sections: "Summary" and "Log output".

**Summary**

Code SHA-256	xLN61YszDgIYBIRyEfH4dNqvZE3ZKC18XC58P5BPXul=
Request ID	13cbc61c-23af-11e7-8598-0552d0ec5bf0
Duration	237.84 ms

**Log output**

The area below shows the logging calls in your code. These correspond to a single row within the CloudWatch log group corresponding to this Lambda function. [Click here](#) to view the CloudWatch log group.

```
| START RequestId: 13cbc61c-23af-11e7-8598-0552d0ec5bf0 Version: $LATEST  
| END RequestId: 13cbc61c-23af-11e7-8598-0552d0ec5bf0  
| REPORT RequestId: 13cbc61c-23af-11e7-8598-0552d0ec5bf0 Duration: 237.84 ms Billed Duration:
```

# TESTING

- Firebase
  - Local development server
  - Deploy/test cycle is slow
  - Console only shows logged information

Lambda > Functions > helloWorld

Qualifiers ▾ Test Actions ▾

Code Configuration Triggers Tags Monitor

Code entry type Edit code inline

```
1 from __future__ import print_function
2
3 import json
4
5 print('Loading function')
6
7
8 def lambda_handler(event, context):
9     #print("Received event: " + json.dumps(event, indent=2))
10    print("value1 = " + event['key1'])
11    print("value2 = " + event['key2'])
12    print("value3 = " + event['key3'])
13
14    return {"statusCode": 200, \
15            "headers": {"Content-Type": "application/json"}, \
16            "body": {"message": "hello world"}}
17
18    #return event['key1'] # Echo back the key from the trigger
19    #raise Exception('Something went wrong')
20
```

# PRICING

# PRICING

- Pricing metrics
  - Number of invocations
  - Duration of invocation
  - Hardware

Lambda > Functions > helloWorld

Qualifiers ▾ Test Actions ▾

Code Configuration Triggers Tags Monitor

Code entry type Edit code inline

```
1 from __future__ import print_function
2
3 import json
4
5 print('Loading function')
6
7
8 def lambda_handler(event, context):
9     #print("Received event: " + json.dumps(event, indent=2))
10    print("value1 = " + event['key1'])
11    print("value2 = " + event['key2'])
12    print("value3 = " + event['key3'])
13
14    return {"statusCode": 200, \
15            "headers": {"Content-Type": "application/json"}, \
16            "body": {"message": "hello world"}}
17
18    #return event['key1'] # Echo back the key provided in the event.
19    #raise Exception('Something went wrong.')
20
```

# PRICING

- Cloud Functions
  - \$.40 per million (2M free)
- Lambda
  - \$.20 per millions (1M free)

Lambda > Functions > helloWorld

Qualifiers ▾ Test Actions ▾

Code Configuration Triggers Tags Monitor

Code entry type Edit code inline

```
1 from __future__ import print_function
2
3 import json
4
5 print('Loading function')
6
7
8 def lambda_handler(event, context):
9     #print("Received event: " + json.dumps(event, indent=2))
10    print("value1 = " + event['key1'])
11    print("value2 = " + event['key2'])
12    print("value3 = " + event['key3'])
13
14    return {"statusCode": 200, \
15            "headers": {"Content-Type": "application/json"}, \
16            "body": {"message": "hello world"}}
17
18    #return event['key1'] # Echo back the key provided in the event.
19    #raise Exception('Something went wrong')
20
```

# ODDS AND ENDS

# STATUS AND DOCUMENTATION

- Cloud functions
  - In beta
  - Up and coming
  - Updated documentation

## CLOUD FUNCTIONS BETA

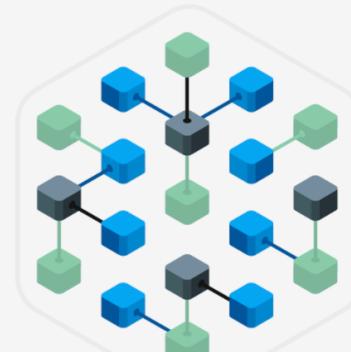
A serverless environment to build and connect cloud services

(...) [VIEW DOCUMENTATION](#)

[VIEW CONSOLE](#)

### Serverless Applications on Google's Infrastructure

Cloud computing has made possible fully serverless models of computing where logic can be spun up on-demand in response to events originating from anywhere. Construct applications from bite-sized business logic billed by the nearest 100 milliseconds, only while your code is running. Serve users from zero to planet-scale, all without managing any infrastructure.



### Microservices Over Monolith

Developer agility comes from building systems with functionality focused on doing one thing well. Break down monolithic services at the level of a single function, not entire VMs.

## STATUS AND DOCUMENTATION

- Lambda
  - Generally available
  - Documentation is outdated in many circumstances
  - Stack Overflow has correct answers to old problems



### AWS Lambda

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

[Get started with AWS Lambda](#)

## STATUS AND DOCUMENTATION

- App Engine
  - More control
  - 3rd party modules
  - Any language

## GOOGLE APP ENGINE

Build scalable web and mobile backends in any language on Google's infrastructure

 [VIEW DOCUMENTATION](#)

## App Engine for All

Build modern web and mobile applications on an open cloud platform: bring your own language runtimes, frameworks, and third party libraries. Google App Engine is a fully managed platform that completely abstracts away infrastructure so you focus only on code. Go from zero to planet-scale in minutes—see why some of today's most successful companies power their applications on App Engine.

### For All Language Communities

Out of the box, App Engine supports Node.js, Java, Ruby, C#, Go, Python, and PHP. Developers from these language communities can be productive immediately in a familiar environment without adding code.

# REALM

REALM

# Solve the toughest challenges in mobile app dev

- Mobile first database
  - Cross platform
  - Full "Platform" for mobile app backends
- **Powerful offline-first features**
  - **Mobile-savvy REST API integrations**
  - **Seamless mobilization of legacy data**
  - **Realtime performance and live collaboration**
  - **Advanced caching and edge compute**

[Learn More](#)

IN THE BEGINNING...

# REALM

- Initially on device only
- Billed as a CoreData alternative
  - But easy to use
- Faster than SQLite

## Realm Mobile Database

Loved by developers and more than a billion users, Realm Mobile Database is fast, easy to use, open source, and totally free.



# REALM

- UI Components tuned for the Realm database
  - UITableViewController/  
NSFetchedResultsController

## Realm Mobile Database

Loved by developers and more than a billion users, Realm Mobile Database is fast, easy to use, open source, and totally free.



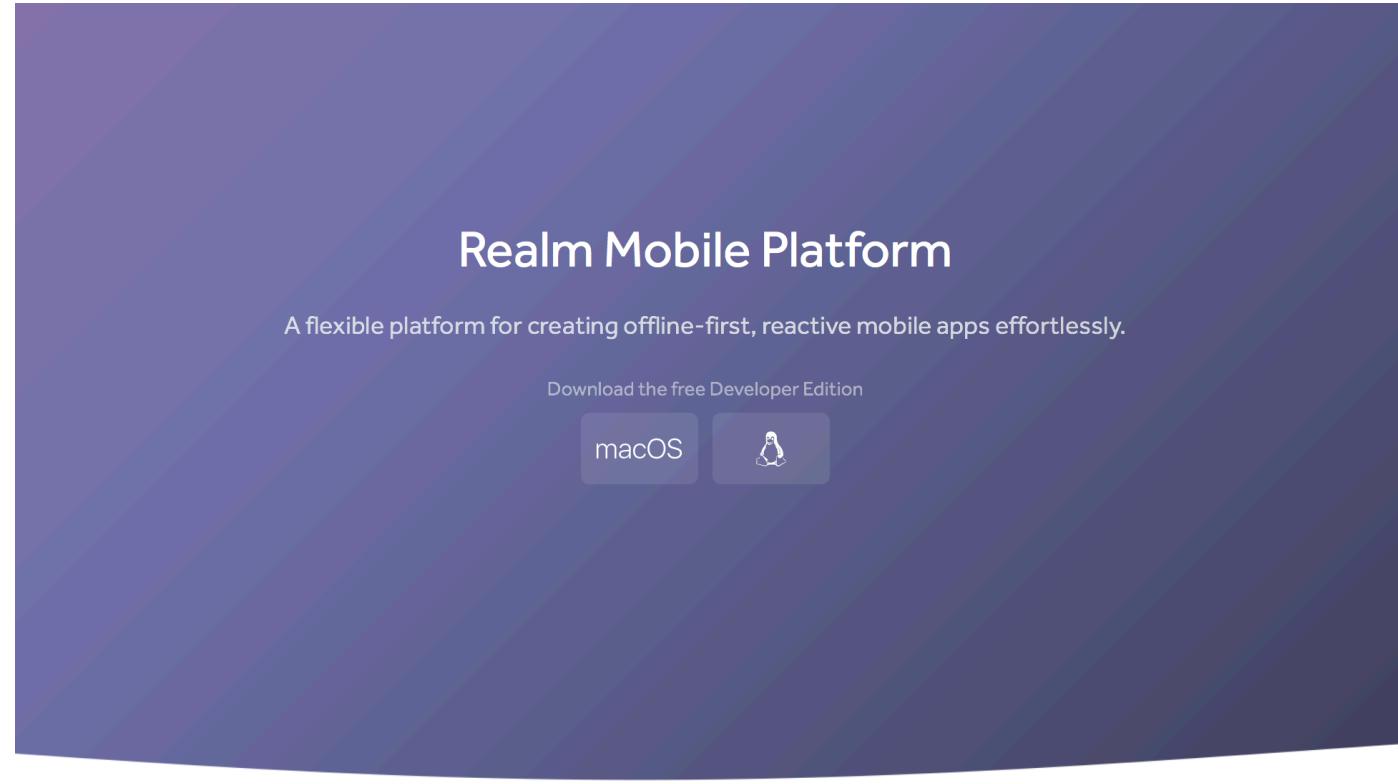
# REALM

- Simple **local** object persistence
- Thread safe
- Encryption built in

```
class Dog: Object {  
    dynamic var name = ""  
    dynamic var age = 0  
}  
  
let dog = Dog()  
dog.name = "Rex"  
dog.age = 1  
  
let realm = try! Realm()  
try! realm.write {  
    realm.add(dog)  
}  
  
let pups = realm.objects(Dog.self).filter("age < 2")
```

# REALM

- On-Device limitation until...



The screenshot shows the official website for the Realm Mobile Platform. The header features the word "REALM" in large, white, sans-serif letters. Below the header is a large, dark purple rectangular area containing the text "Realm Mobile Platform" in white, followed by a subtitle "A flexible platform for creating offline-first, reactive mobile apps effortlessly." in a smaller white font. A call-to-action button labeled "Download the free Developer Edition" is present, with two download links: "macOS" and "Linux".

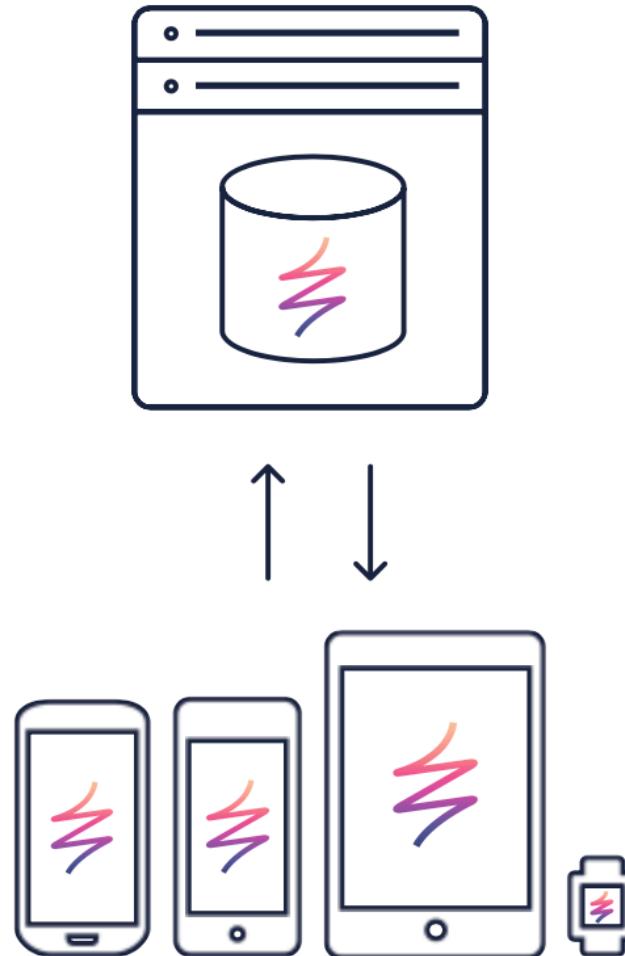
The perfect backend for the next generation of reactive mobile apps

The Realm Mobile Platform delivers automatic and seamless realtime data sync and powerful event handling between server and devices. You never need to think



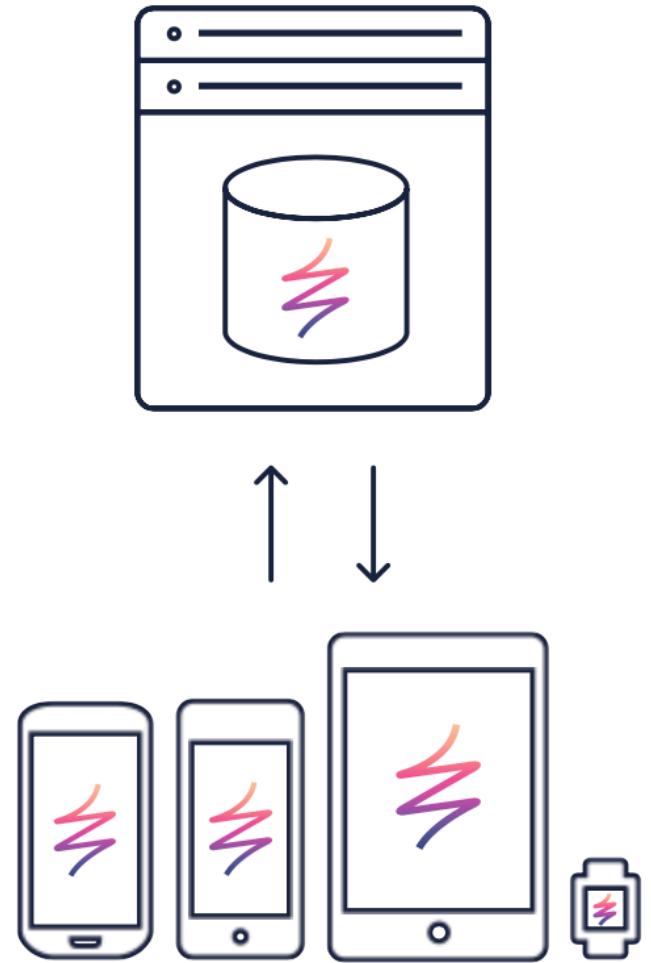
# REALM

- Sync engine that runs on any cloud
- Nginx node.js server
- Public or private clouds



# REALM

- Local cache and sync
- Live objects with observer model (like Firebase)
- Subscriptions (like CloudKit)



REALM

# Serverless Logic with Realm: Introducing Realm Functions

Realm Team

May 23 2017

- But limitation of no server side logic, until....

 Transcript



About the Speaker



Today we're announcing Realm Functions, a new part of Realm that makes building server-side functionality a lot easier for mobile developers. Now, you can make server-side features without listing backend developers, plus you get all the benefits of building on top of the Realm Mobile Platform: you don't need to add another endpoint to a server, and then write the serialization and networking code that would let you connect with it. You just connect your app to Realm, write a Realm Function in your web dashboard, and watch your code execute reactively as data streams in. Today's release is a beta, and it's available today to everyone, whether you're building an app in an enterprise-scale team or for a small side project.

# REALM

The image shows two screenshots demonstrating the use of Realm Functions. On the left, a screenshot of the 'Realm Object Server v1.7.0-beta7' interface on a Mac shows a code editor with the following JavaScript code:

```
15 var task = tasks[taskIndex];
16 console.log("New task received: " + change_event.path);
17 // get date from wit.ai
18 // probably use this https://wit.ai/docs/http/20160526#get--message-link
19 // node-wit: https://github.com/wit-ai/node-wit
20 witClient.message(task.text, {}).then((data) => {
21   console.log("Response received from wit: " + JSON.stringify(data));
22   var dateTime = data.entities.datetime[0];
23   if (!dateTime) {
24     console.log("Couldn't find a date.");
25     return;
26   }
27   console.log("Isolated calculated date: " + dateTime.value);
28   // to write the date, we'll have to add a date property on the client and migi
29   realm.write(function() {
30     task.date = new Date(dateTime.value);
31   });
32 }
33 .catch(console.error);
34 }
35 };
```

The right screenshot shows an iPhone 7 Plus displaying a list of tasks under 'My Tasks'. The tasks are:

- Test Realm Functions At 5pm
- Listen To Music
- Clean The Office
- Learn How To Code
- Finish Reading Book
- Pick Up Groceries
- Go To The Gym

The task 'Clean The Office' has a cursor icon over it, indicating it is selected.

- <https://news.realm.io/news/serverless-logic-with-realm-introducing-realm-functions/>

# REALM PLATFORM

REALM  
PLATFORM

# Realm Platform 2.0

Everything you love about the Realm Platform just got better:

- Database
- Platform
- Studio

- **New Realm Studio: easily manage your Realms, users, and config**
- **Easy and simple NPM install**
- **Improved, fully pluggable authentication system**
- **A host of stability, performance, and usability enhancements**

[Watch the webinar](#)

or

[try 2.0 today](#)

## REALM PLATFORM

- Realm Object Server (ROS)
  - Realtime sync objects (no REST API)
  - Objects are live (no transport layer)
  - No networking code (that you write)



# REALM PLATFORM

```
curl -s https://raw.githubusercontent.com/realm/realm-object-server/master/install.sh | bash  
  
ros init my-app  
  
cd my-app/  
npm start
```

- Supports Ubuntu, Mac, Windows
  - Public/private clouds
- Run install script on server to download and configure

# REALM PLATFORM

- Supports Ubuntu, Mac, Windows
- Run install script on server to download and configure

DOES MORE THAN SYNC



## Realm Database

Our fast and reactive database is superior to SQLite-based alternatives as an embedded “live object” database on the device. And when you connect it to Realm Object Server, it becomes a distributed database providing automatic, realtime data synchronization.

## Realm Sync

At the heart of Realm Object Server, automatically synchronizing data objects across all devices and the servers in realtime, is Realm Sync. It handles conflict resolution and offline states seamlessly — and your data is safe with TLS/SSL and AES-256 encryption.

## Realm Studio

Functioning as your dashboard and your cockpit, Realm Studio gives you control over your data, platform functions, users, and configuration. Featuring an efficient, task-oriented UI, it's built on Electron and it works across every major platform.

## Realm Connect

Connect to Realm Object Server, and Realm Connect converts existing REST APIs and data sources to live objects, freeing you to focus on features rather

## Event handling

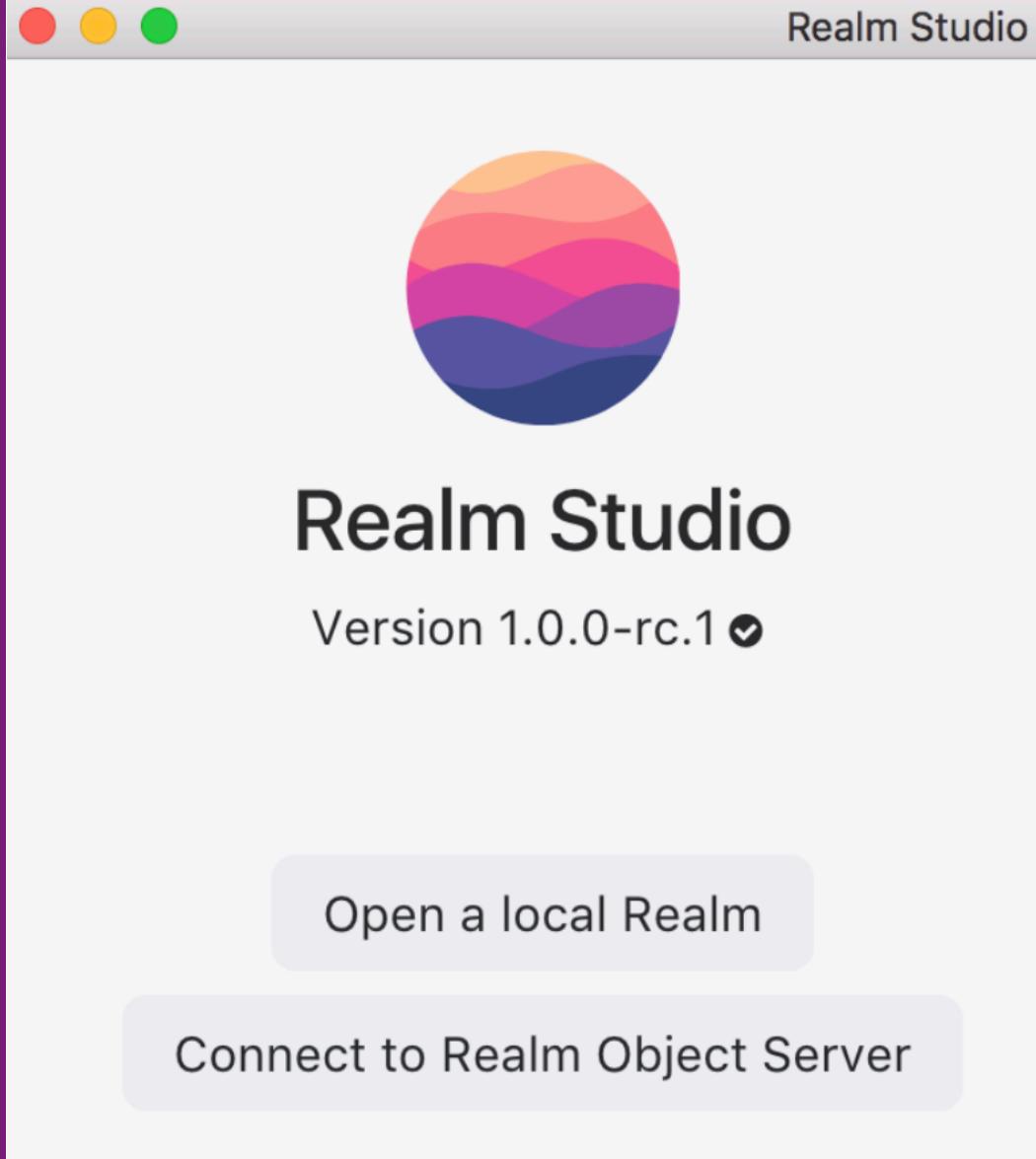
Realm Object Server's event handling functionality means you can easily build server-side features with simple JavaScript. When synced data

## Permissions and authentication

Log in users with our authentication systems, or customize authentication to work with your existing

## REALM PLATFORM

- Realm Studio app
  - Manage server data
    - Add, delete, edit
  - View logs



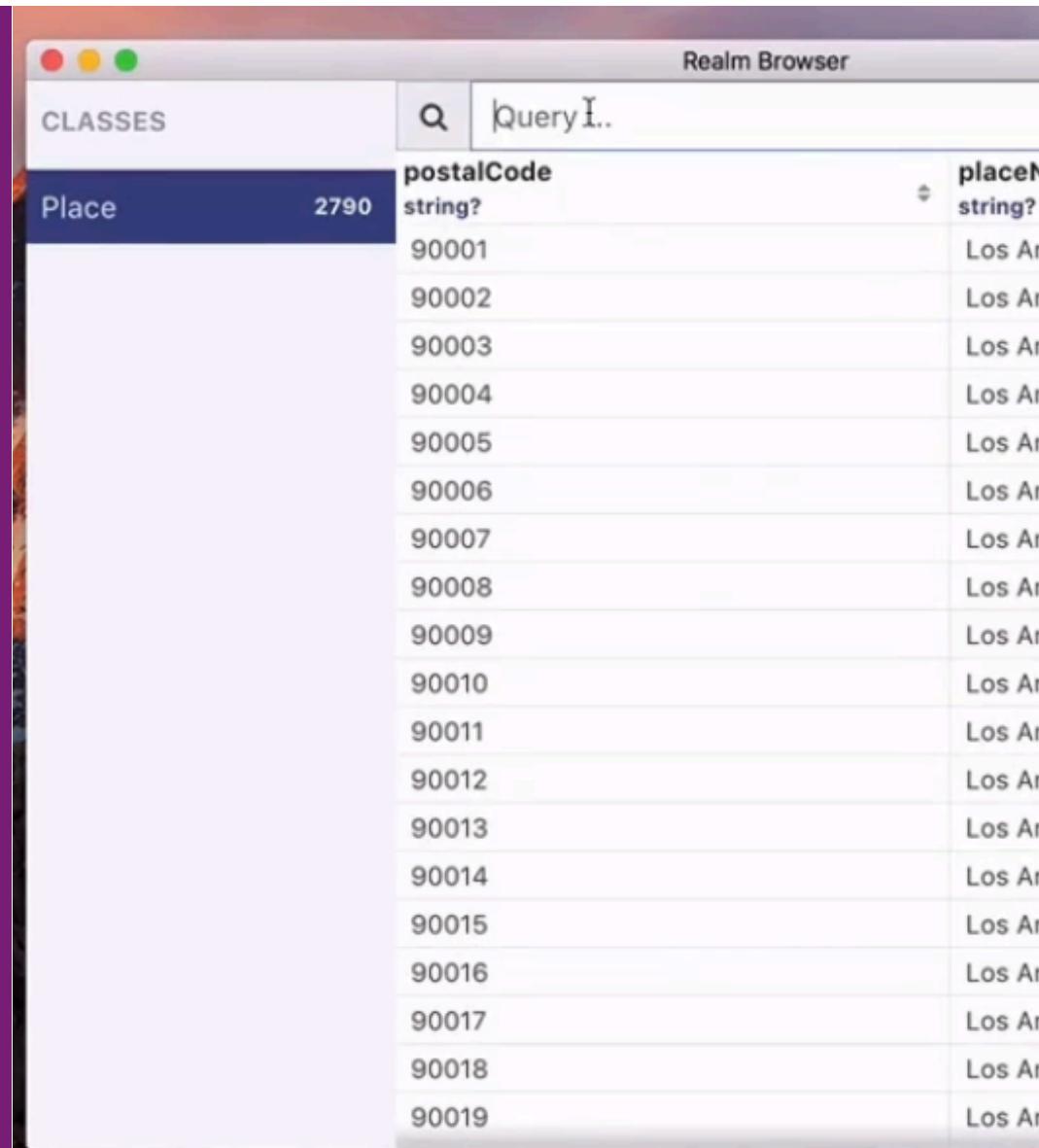
# REALM PLATFORM

Connected to http://localhost:9080

ID	ROLE	# REALMS
8cebf082313c701a821c0d1b56ecf14	Administrator	0

# REALM PLATFORM

- How does the sync engine work?
  - App Engine, explicit calls
  - Firebase 🤖
  - CloudKit, subscriptions



The screenshot shows the Realm Browser application interface. At the top, there's a navigation bar with 'Realm Browser' and a search bar labeled 'Query...'. Below the header, the word 'CLASSES' is displayed. A table lists objects of the 'Place' class. The first column shows the object ID, the second column shows the count '2790', and the third column shows the 'postalCode' field. The data in the 'postalCode' column is as follows:

Object ID	Count	postalCode
90001		Los Angeles, CA
90002		Los Angeles, CA
90003		Los Angeles, CA
90004		Los Angeles, CA
90005		Los Angeles, CA
90006		Los Angeles, CA
90007		Los Angeles, CA
90008		Los Angeles, CA
90009		Los Angeles, CA
90010		Los Angeles, CA
90011		Los Angeles, CA
90012		Los Angeles, CA
90013		Los Angeles, CA
90014		Los Angeles, CA
90015		Los Angeles, CA
90016		Los Angeles, CA
90017		Los Angeles, CA
90018		Los Angeles, CA
90019		Los Angeles, CA

# REALM PLATFORM

- Realm uses notifications that are sent when it observes a change
  - Realms
  - Collections
  - Objects

```
// Observe Realm Notifications
let token = realm.observe { notification, realm in
    viewController.updateUI()
}

// later
token.invalidate()
```

# REALM PLATFORM

- Collections

```
// Observe Results Notifications
notificationToken = results.observe { [weak self] (changes: RealmCollectionChange) in
    guard let tableView = self?.tableView else { return }
    switch changes {
        case .initial:
            // Results are now populated and can be accessed without blocking the UI
            tableView.reloadData()
        case .update(_, let deletions, let insertions, let modifications):
            // Query results have changed, so apply them to the UITableView
            tableView.beginUpdates()
            tableView.insertRows(at: insertions.map({ IndexPath(row: $0, section: 0)}),
                                with: .automatic)
            tableView.deleteRows(at: deletions.map({ IndexPath(row: $0, section: 0)}),
                                with: .automatic)
            tableView.reloadRows(at: modifications.map({ IndexPath(row: $0, section: 0)}),
                                with: .automatic)
            tableView.endUpdates()
        case .error(let error):
            // An error occurred while opening the Realm file on the background worker thread
            fatalError("\(error)")
    }
}
```

# REALM PLATFORM

- Objects

```
class StepCounter: Object {
    @objc dynamic var steps = 0
}

let stepCounter = StepCounter()
let realm = try! Realm()
try! realm.write {
    realm.add(stepCounter)
}
var token : NotificationToken?
token = stepCounter.observe { change in
    switch change {
        case .change(let properties):
            for property in properties {
                if property.name == "steps" && property.newValue as! Int > 1000 {
                    print("Congratulations, you've exceeded 1000 steps.")
                    token = nil
                }
            }
        case .error(let error):
            print("An error occurred: \(error)")
        case .deleted:
            print("The object was deleted.")
    }
}
```

# REALM PLATFORM

- Realm takes advantage of Key-Value-Observing (KVO)
- KVO is a mechanism enables objects to register for asynchronous notifications driven by changes in another object's properties

```
import Foundation

//: # Santa object subclass of NSObject #
@objcMembers // Activate objective-c's introspection capabilities
class Santa: NSObject {

    // The dynamic declaration modifier is required whenever you
    // make use of Objective-C's dynamism
    dynamic var string: String

    override init() {
        string = "🎅"
        super.init()
    }
}

//: # Print original Santa #
let santa = Santa()
print("Original: \(santa.string)")

//: # Set up an observer #
// ". " is shorthand for Santa (in this case); new Key Path in Swift
let observation = santa.observe(\.string) { (santa, change) in
    print("We observed a change in a property. The new value is \(change.newValue)")
}

//: # Make an observable change #
// We are changing this...watch what happens
print("Getting ready to change....")
santa.string = "santa"
```

# REALM PLATFORM

```
import Foundation

//: # Santa object subclass of NSObject #
@objcMembers // Activate objective-c's introspection capabilities
class Santa: NSObject {

    // The dynamic declaration modifier is required whenever you need to
    // make use of Objective-C's dynamism
    dynamic var string: String

    override init() {
        string = "🎅"
        super.init()
    }
}

//: # Print original Santa #
let santa = Santa()
print("Original: \(santa.string)")
```



# REALM PLATFORM

- When a change happens on device, it triggers update to server
- When a change happens on server, it triggers update on device

```
import Foundation

//: # Santa object subclass of NSObject #
@objcMembers // Activate objective-c's interface
class Santa: NSObject {

    // The dynamic declaration modifier is required
    // make use of Objective-C's dynamism
    dynamic var string: String

    override init() {
        string = "🎅"
        super.init()
    }
}

//: # Print original Santa #
let santa = Santa()
print("Original: \(santa.string)")

//: # Set up an observer #
// ".observe" is shorthand for observe(in:)
let observation = santa.observe(\.string) { value in
    print("We observed a change in a property")
}

//: # Make an observable change #
// We are changing this...watch what happens
print("Getting ready to change....")
santa.string = "santa"
```

# REALM PLATFORM

- Similar to CloudKit subscriptions in that changes that match a predicate trigger a notification to be sent to all clients
  - Very limited amount of work setting it up and handling the notifications in CloudKit

```
subclass of NSObject #  
private objective-c's introspection capability  
ct {  
  
declaration modifier is required whenever you want  
objective-C's dynamism  
g: String  
  
    . Santa #  
  
    santa.string")  
  
server #  
    l for Santa (in this case); new Key Path in  
    Santa.observe(\.string) { (santa, change) in  
        l a change in a property. The new values is  
  
        variable change #  
        .s...watch what happens  
        to change....")  
        :a"
```

## REALM PLATFORM

- <https://academy.realm.io/posts/learning-path-build-a-realtime-swift-app-with-realm/>



### Build a Realtime Swift App with Realm

Get hands-on with the Realm Platform, build a realtime Santa-tracking app, and learn ne...

# REALM PLATFORM

- <https://academy.realm.io/posts/learning-path-build-a-realtime-swift-app-with-realm/>



global tour!



data changes.



## Track Santa with Realm: Part 1

Learn how to build a Swift app with the Realm Platform by tracking Santa on his

## Track Santa with Realm: Part 2

Build on your app by completing the data models and making your UI reactive to

## Track Santa with Realm: Part 3

TO REALM OR NOT TO  
REALM

# TO REALM

- Realm checks off all the boxes for a mobile backend
  - Local persistence
  - Sync
  - SDK with networking
  - Authentication

CLASSES	Q	Query...
Place	2790	
	postalCode	placeName
	string?	string?
	90001	Los Angeles
	90002	Los Angeles
	90003	Los Angeles
	90004	Los Angeles
	90005	Los Angeles
	90006	Los Angeles
	90007	Los Angeles
	90008	Los Angeles
	90009	Los Angeles
	90010	Los Angeles
	90011	Los Angeles
	90012	Los Angeles
	90013	Los Angeles
	90014	Los Angeles
	90015	Los Angeles
	90016	Los Angeles
	90017	Los Angeles
	90018	Los Angeles
	90019	Los Angeles

# REALM PLATFORM

## 2. Handling the data on the server

This is the cool part. Where the magic happens. Granted, this example isn't very *exciting* magic, but hey, it's magic.

Server-side data access is only available through the Professional or Enterprise Editions of the Realm Platform. You can read more about those, or sign up for a free trial, on the [Pricing page](#).

So I'm going to gloss right over setting up your Realm Obj...  
that. For this tutorial, I'm assuming you have

We'll open our `index.js` file:

```
'use strict';
```

Syncing My First Data With Realm Object Server

Since there is no data yet in the server, let's use `realm-js` to create a sample data set.  
To complete this step, you will need to sign up for our Professional Edition trial [here](#).

check out the

how to do



# TO REALM

- Professional edition
  - Event handling (functions)
  - Server access
  - Data migration tools
  - Horizontal Scaling?

Realm Browser

CLASSES	Q	Query...
Place	2790	postalCode string? placeName string?
		90001 Los Angeles
		90002 Los Angeles
		90003 Los Angeles
		90004 Los Angeles
		90005 Los Angeles
		90006 Los Angeles
		90007 Los Angeles
		90008 Los Angeles
		90009 Los Angeles
		90010 Los Angeles
		90011 Los Angeles
		90012 Los Angeles
		90013 Los Angeles
		90014 Los Angeles
		90015 Los Angeles
		90016 Los Angeles
		90017 Los Angeles
		90018 Los Angeles
		90019 Los Angeles

# REALM PLATFORM

## Developer Edition

**Free**

Enable realtime two-way data sync for mobile devices and servers.

No cost to use, even in production for commercial use cases. [Read the terms.](#)

[Get the download](#)

## Professional Edition

**Free Trial**

Powerful features like simplified integrations, event handling, server-side data access.

14-day free trial; pricing starts at \$1,750 per month.

[Download free trial](#)



## Enterprise Edition

**Free Demo**

Enterprise scale, availability, and integration with existing infrastructure/third-party APIs.

Pricing scales with use case.

[Contact us for demo](#)



# TO REALM

- A hybrid of Firebase and CloudKit
  - Less blackbox
  - Requires some "fiddling" on the server
- Privacy (end-to-end control)
- Cross-platform

Realm Platform	
Developer Edition	Professional Edition
<b>Free</b>	<b>Free Trial</b>
Enable realtime two-way data sync for mobile devices and servers.	Powerful features like simplified integrations, event handling, server-side data access.
No cost to use, even in production for commercial use cases. <a href="#">Read the terms.</a>	14-day free trial; pricing starts at \$1,750 per month.
<a href="#">Get the download</a>	<a href="#">Download free trial</a>
<b>Object Database</b>	<b>Object Database</b>
✓	✓
<b>On-premises or Public Cloud</b>	<b>On-premises or Public Cloud</b>
✓	✓
<b>Cross Platform</b>	<b>Cross Platform</b>
Android, iOS, Xamarin, and JavaScript	Android, iOS, Xamarin, and JavaScript
<b>Data Encryption</b>	<b>Data Encryption</b>
AES-256 at rest; SSL/TLS in flight	AES-256 at rest; SSL/TLS in flight
<b>Offline-first Functionality</b>	<b>Offline-first Functionality</b>
✓	✓

# CLASS IN REVIEW

# COURSE DESIGN

# COURSE DESIGN

- Today an app is not enough
  - Complexity
  - Interconnected
  - User expectations
  - Insights/analytics
- Provide exposure to technologies to complete mobile applications

This was the goal

# COURSE DESIGN



## COURSE DESIGN

- Current state of the field
  - Technologies come and go
  - Consolidation/duplication of services
  - DIY mentality
  - Industry proving ground doesn't fit with the narrative



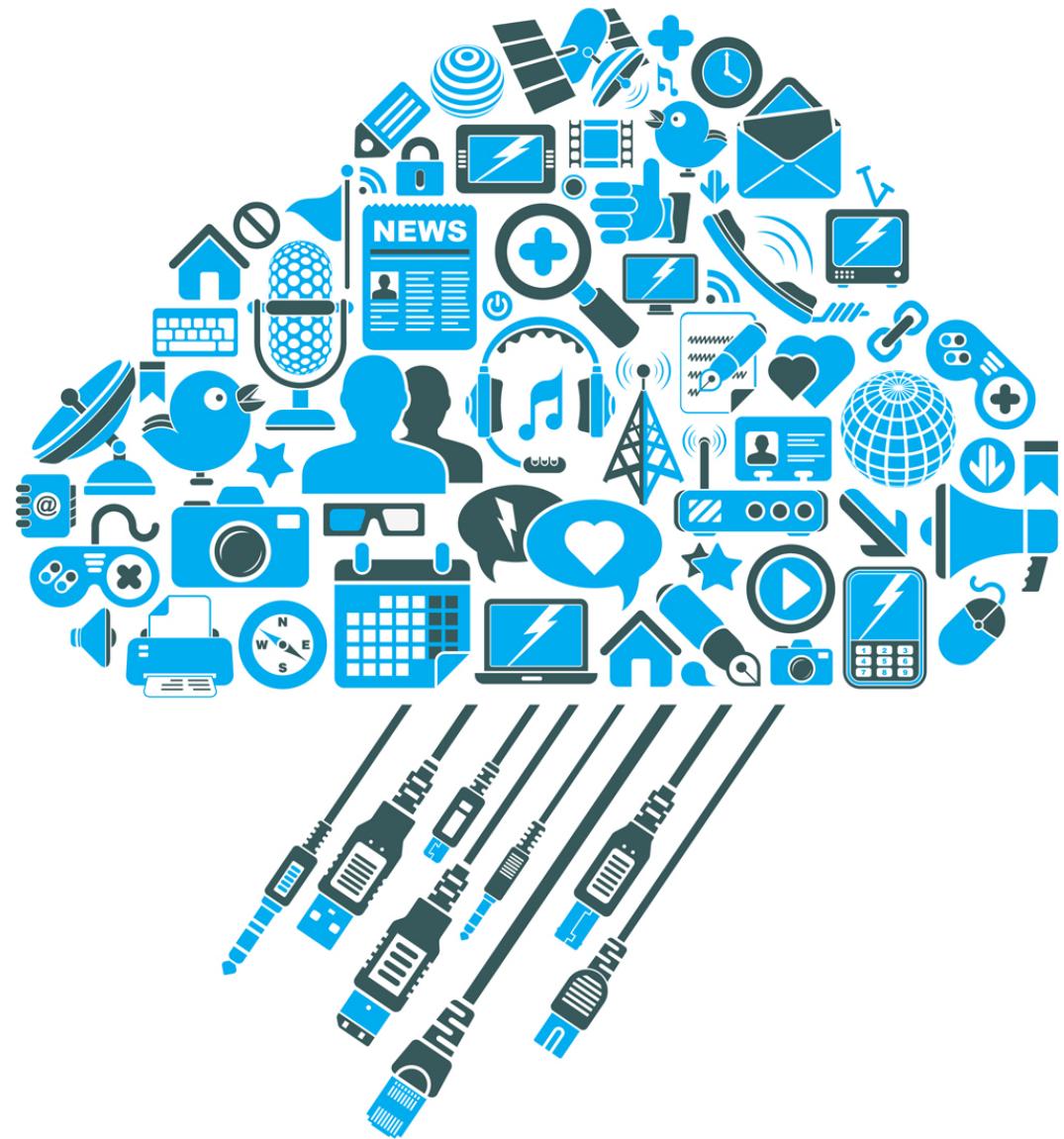
**Google App  
Engine**



# WHERE WE STARTED

## BACKENDS FOR MOBILE APPLICATION

- Backends is a specific utilization of cloud computing



# BACKENDS FOR MOBILE APPLICATION

- Platforms
  - Provides on demand services
  - Pay for use and as needed, elastic
  - Scale up and down in capacity and functionalities
  - The hardware and software services are available to general public, enterprises, corporations and businesses markets



# BACKENDS FOR MOBILE APPLICATION

## Common Characteristics:

**Massive Scale**

**Resilient Computing**

**Homogeneity**

**Geographic Distribution**

**Virtualization**

**Service Orientation**

**Low Cost Software**

**Advanced Security**

## Essential Characteristics:

**On Demand Self-Service**

**Broad Network Access**

**Rapid Elasticity**

**Resource Pooling**

**Measured Service**

# BACKENDS FOR MOBILE APPLICATION

- Cloud service models
  - Software as a service (SaaS)
    - Adobe Creative Cloud
    - Platform as a service (PaaS)
  - Infrastructure as a service (IaaS)
    - Compute Engine

## BACKENDS FOR MOBILE APPLICATION

- Different Cloud Computing service layers

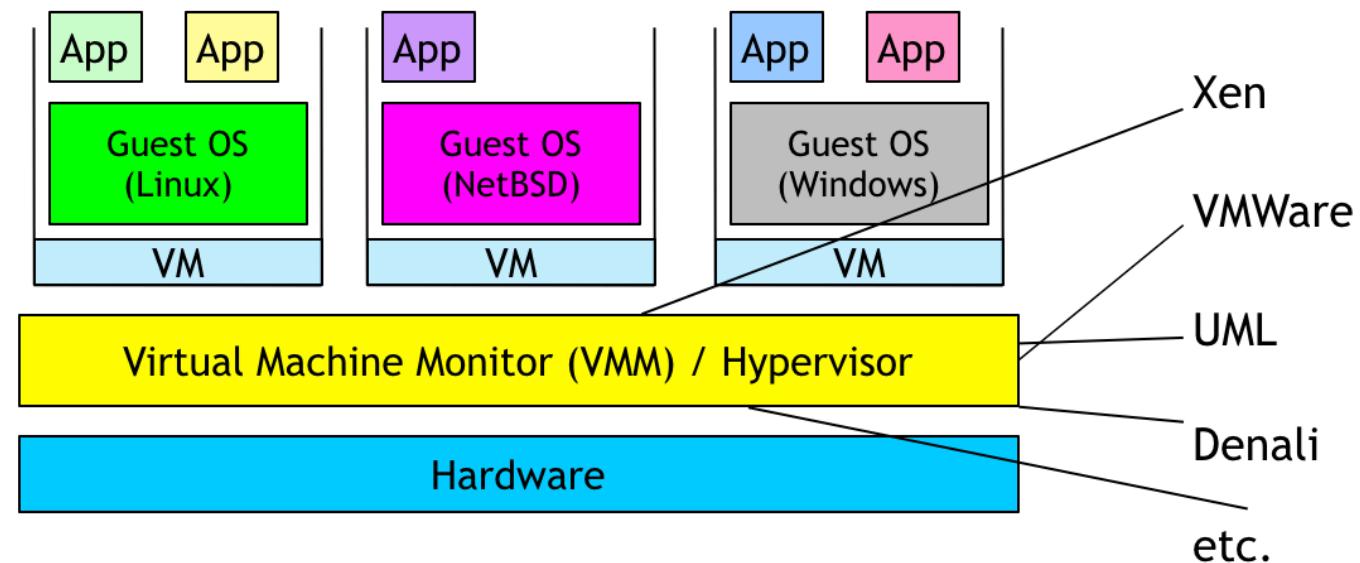
Services	Description
Services	Services - Complete business services such as PayPal, OpenID, OAuth, Google Maps, Alexa
Application	Application - Cloud based software that eliminates the need for local installation such as Google Apps, Microsoft Online
Development	Development - Software development platforms used to build custom cloud based applications (PAAS & SAAS) such as SalesForce
Platform	Platform - Cloud based platforms, typically provided using virtualization, such as Amazon ECC, Sun Grid
Storage	Storage - Data storage or cloud based NAS such as CTERA, iDisk, CloudNAS
Hosting	Hosting - Physical data centers such as those run by IBM, HP, NaviSite, etc.

**Application Focused**

**Infrastructure Focused**

## BACKENDS FOR MOBILE APPLICATION

- VM (virtual machine) technology allows multiple virtual machines to run on a single physical machine
- Abstraction of a physical machine



# BACKENDS FOR MOBILE APPLICATION

- Advantages
  - Performance, cost, access, collaboration, reliability, bought expertise, security
- Disadvantages
  - Need a network, give up control, monopolization of industry, security



WHERE WE ENDED

# BACKENDS FOR MOBILE APPLICATION

- Platforms
  - App Engine
  - Firebase
  - iCloud (CloudKit)
  - Hosted VMs with Swift Server Frameworks
  - Realm



# BACKENDS FOR MOBILE APPLICATION

- App Engine
  - The most complete offering
  - Flexibility without (too much) complexity
  - Can do everything..plus and a minus
  - Local development environment
  - All client work is up to developer



# BACKENDS FOR MOBILE APPLICATION

- Assignment 2

- Use app engine to create a photo timeline app
- Designed CRUD API
- Basic authentication
- Use Google Vision API

## Assignment 2 Part 2 Mobile Analytics Platform

In this part of the assignment, you will create a mobile analytics platform to collect information about how customers are using your application. The platform should collect and aggregate the following information:

- **User Session.** A user session is defined as the amount of time an application is continuously active. For example, if a user opens the app uses it for 30 seconds and then switches to another app, this would be a single session of length 30 seconds. Later that day, if the user opens the app and uses it for 180 seconds, that would be a second user session of 180 seconds.
- **User Events.** A user even is recorded when the user performs a specific action in the application. For example, you may choose to record button taps, segue to specific view controllers, or the numbers of times that a user reloaded a page. *For this application and simplicity, we will only be concerned with a button tap user event.*
- **Touch Heat Map.** Our app is designed for single handed use and our UI/UX team is considering how the screens size of the “plus” phones is effecting the usability of our app. We want to collect all the touches that a user makes in a given session.

Every day at 10pm, you should send an email to the developer with a summary of the apps usage statistics. The information in the email should include the following:

- **Daily Active Users.** This represents the number of unique users that used your app on a given day.
- **Daily New Users.** The number of new users for a given day.
- **Monthly Active Users.** The number of users for the current month

## BACKENDS FOR MOBILE APPLICATION

- Assignment 3
  - Created a mobile analytics backend on App Engine
  - Microservice architecture
  - Store data and server logic
  - Cron job to send email summary

## Mobile Analytics Platform

In this part of the assignment, you will create a mobile analytics platform to collect information about how customers are using your application. The platform should collect and aggregate the following information:

- **User Session.** A user session is defined as the amount of time an application is continuously active. For example, if a user opens the app uses it for 30 seconds and then switches to another app, this would be a single session of length 30 seconds. Later that day, if the user opens the app and uses it for 180 seconds, that would be a second user session of 180 seconds.
  - **User Events.** A user event is recorded when the user performs a specific action in the application. For example, you may choose to record button taps, segment specific view controllers, or the numbers of times that a user reloaded a page. *For this application and simplicity, we will only be concerned with a button tap user event.*
  - **Touch Heat Map.** Our app is designed for single handed use and our UI/UX team is considering how the screen size of the "plus" phones is effecting the usability of our app. We want to collect all the touches that a user makes during a given session.
- Every day at 10pm, you should send an email to the developer with a summary of the apps usage statistics. The information in the email should include the following:
- **Daily Active Users.** This represents the number of unique users that used the app on a given day.

# BACKENDS FOR MOBILE APPLICATION

- Firebase
  - Ease of use, broad spectrum of offerings
  - Extensible with GCP
  - Mobile first libraries
  - Unique mobile-first offerings (invites, authentication, etc.)
  - Structuring data is not-intuitive



## BACKENDS FOR MOBILE APPLICATION

- Assignment 4
  - Chat app using Firebase
  - Dynamic links with invites
  - Structuring a database with
  - Cloud functions
  - Integrate with mobile app

# Assignment By Invitation Only An Exclusive Chat Application

In this assignment, you will create a group chat application. You will implement many features available through the Firebase platform. You will implement the common functionality of group chat applications to allow users to communicate with each other.

## Overall Application Behavior

All users will have to be authenticated with Google authentication. After signing in, users should be presented with a list of groups they belong to. Navigating to one of these groups should take them to a group chat interface for that application. Users should be allowed to post a new message to the group.

Chat groups are **invite-only** and the only way to join one is by being invited by an existing member. Users of the application will be allowed to invite others to join the group, hence becoming the first member, and then be allowed to invite others to join the group.

# BACKENDS FOR MOBILE APPLICATION

- CloudKit
  - Integration with iOS
  - No local storage
  - Finicky 🤯
  - Subscriptions
  - Server to server API



## BACKENDS FOR MOBILE APPLICATION

- Assignment 4
  - Joke of the day using CloudKit
  - Server to server logic offloaded to Google App Engine (👍)
  - Notification tool

### Joke of the Day 😂

In this part of the assignment, you will create a crowd sourced "Joke of the Day" application. The application will be backed by CloudKit with additional logic provided by a microservice running on Google App Engine.

#### Overall Application Behavior

Users will be able to submit a joke through the iOS application. The application will be comprised of both the text of the joke and audio of the joke being read. Users will be able to view and listen to all the jokes and rate them. All users should receive a notification when a new joke is submitted. A cron job will be set up so that they received a notification when a new joke was added to the public database.

Each morning, users will receive a notification informing them of the "Joke of the Day" or the highest rated joke for the previous day. This functionality will be implemented using a cron job service on Google App Engine that communicates with the CloudKit database. At a set time, the service should request the top rated jokes from the previous day and sort them by their rating. The top rated joke will be the "Joke of the Day."

## BACKENDS FOR MOBILE APPLICATION

- VM with Swift on the Server
  - Vapor or Perfect
  - Integration with cloud based PostgreSQL server
  - Deploy Heroku, AWS, Google Cloud



# FINAL THOUGHTS

## FINAL THOUGHTS

**COMPLEXITY**

**APP ENGINE**

**SWIFT ON VM**

**FLEXIBILITY**

**CLOUD FUNCTIONS**

**REALM**

**EASE OF USE**

**FIREBASE**

**CLOUDKIT**

**LIMITATIONS**

## FINAL THOUGHTS

COMPLEXITY



APP ENGINE



EASE OF USE

FIREBASE

CLOUDKIT

CLOUD FUNCTIONS

FLEXIBILITY

LIMITATIONS



## FINAL THOUGHTS

- Is there a clear winner? Nope
- You can make them all do the same thing
- Decision points
  - Client platform
  - Service familiarity
  - Integrations

## FINAL THOUGHTS

- Final decision is probably most determined by the mobile app functionality
  - Keep as much work on the client device
  - Backend fills the gap in functionality

THANK YOU



# THANK YOU

- Feedback on course, content and design
  - Now, later, much later
- Please stay in touch





THE UNIVERSITY OF  
CHICAGO



MPCS 51033 • AUTUMN 2017 • SESSION 10

---

# BACKENDS FOR MOBILE APPLICATIONS