Functions as a Service

(a.k.a. Serverless functions)

Portland State
Computer Science

Functions-as-a-Service

- Recall PaaS
  - Treats servers and computation like electricity (i.e. a commodity consumed on-demand)
  - No VM or container to manage
  - Resources automatically scaled up based on usage
  - Cheapest way to implement microservices that are infrequently used
- FaaS a special case of Platform-as-a-Service
  - Same approach, but done at a function level (versus application one)
- Consists of 2 things
  - An event or trigger
  - A function to run when the event happens
  - e.g. "When an event happens, run this code"
- Sometimes referred to as Internet glue or HTTP duct tape
- A functional programming approach to the cloud
  - No state stored in a function
  - Side-effects pushed out to the edge
  - Allows for greater composability

Example use

- Recall single page application with pre-rendered pages
- Pre-render entire dynamic site (e.g. WordPress, Angular, React) as a single page and forward deploy to client or edge
  - Avoids initial load time of SPA while enabling search engine indexing
- Rendering can be done as a cloud function
  - Render periodically
  - Render upon a change to content

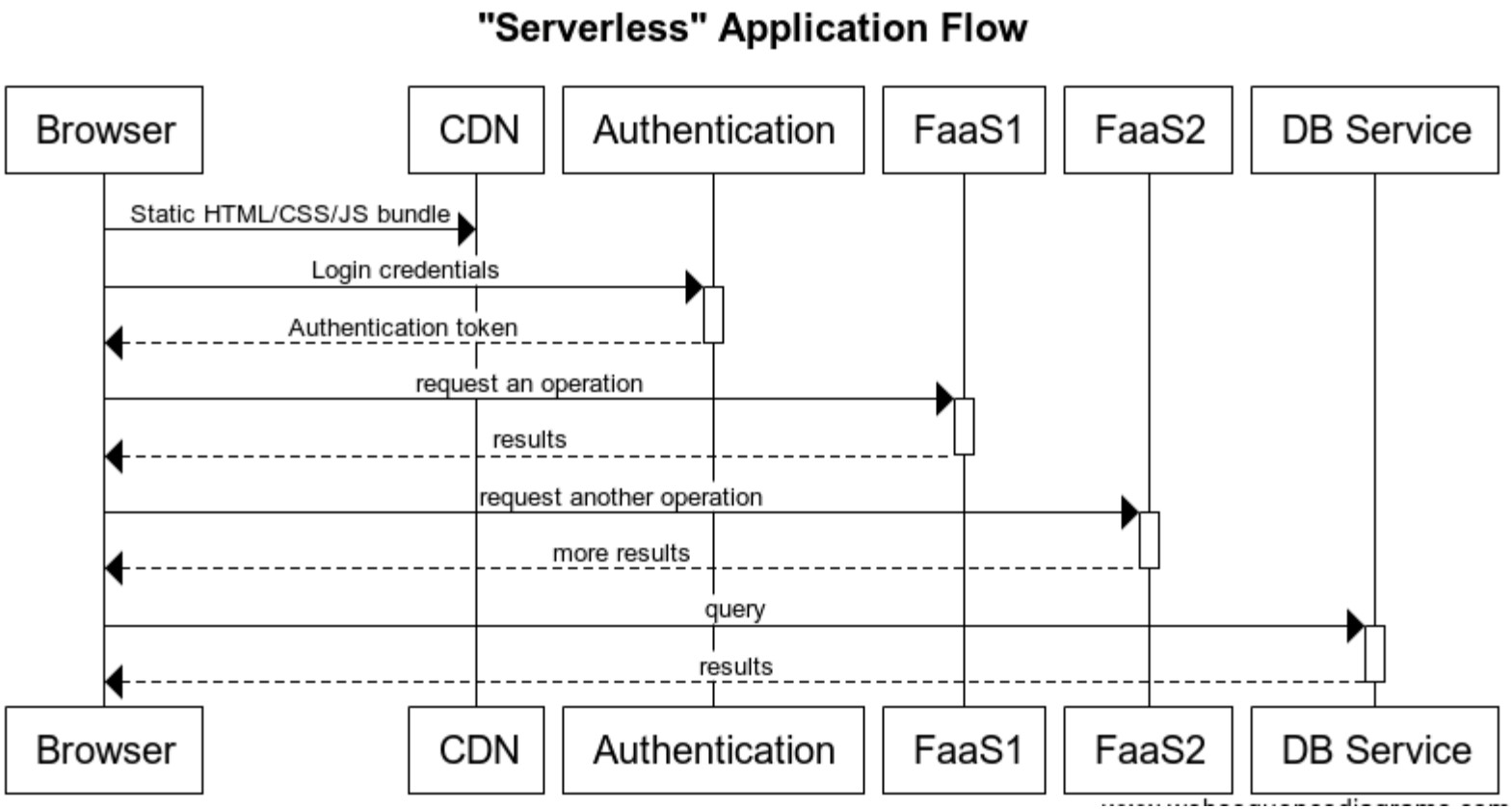
Other use cases

- Perform a speech-to-text conversion when requested (e.g. Alexa)
- Update high-scores of an app/site when database changes
- Run fraud detection or send e-mail welcome upon new user signup
- Ingest sensor data upon new IoT device reading
- Run a function at a particular time (e.g. cron in the cloud)
- Transcode a video (or blur an image) when uploaded by a user (your lab)
- Run a Slack Bot function upon receiving a Slack Slash command (your lab)

Broader patterns

- Managed cloud services and API services often implemented as FaaS
  - Cloud Vision API, Cloud Natural Language Processing API, BigQuery
  - Statistically multiplex at function level versus application/VM level to drive down price
- Good for implementing "Extract, Transform, and Load" pattern (ETL)
  - IoT sensors
- Typically not used to implement entire app
  - Used as glue or for self-contained parts of app
- But…

- Some people do!
  - JAM applications (JavaScript, REST-ful APIs, static Markup)



https://flinthillsgroup.com/web-applications-without-web-servers/

Examples

- AWS Lambda (2014)
- Google Cloud Functions (2016)
- Microsoft Azure Functions (2016)
- Apache OpenWhisk

AWS Lambda

Security?

- Typically, better
  - No persistent malware on them
  - But some assumptions
    - Are the OS and libraries continually patched?
    - Are all resources destroyed when function ends?
- Assumptions often fail
  - Exploitable function exposing underlying run-time (which may have your API keys in them)
  - Caching "hot" functions can allow one to steal credentials if broken
    - Rich Jones – "Gone in 60ms"
- AWS Lambda CTF
  - http://www.lambdashell.com/
- Thunder CTF
- CS 495/595 Serverless Goat, Cloud Goat

Serverless functions issues

- Response times not guaranteed
  - Recently executed functions cached for "hot" operation
  - Idle functions torn down to save resources
  - Cold start for idle functions ~600ms
  - Not good for real-time operations due to unpredictable performance
  - Comparison
    - http://blog.backand.com/serverless-shootout/
- Limited time budget
  - Often implemented on "pre-emptible" VMs
  - Maximum execution on AWS Lambda = 5 min
- Vendor lock-in
- Limited run-time environments*
  - Custom run-times difficult to support

Google Cloud Functions

- Functions as a service running in a standardized, managed environment (mostly Node.js, some Python)
  - User supplies single file defining function and a file listing the packages it requires (e.g. `package.json, requirements.txt`)
  - Runtime compiles function down to native modules via `npm` for deployment (e.g. like Gentoo)
- Function can operate in 2 modes
  - Synchronously (e.g. Implement a REST API that is brought up when an event hits its URL)
  - Asynchronously (e.g. implement a background function that calls back to app when done)

Containers as a Service

Portland State
Computer Science

Containers as a Service

- Serverless platforms
  - Typically implemented with containers
  - Specify an environment
  - Deploy operation creates a container from it and autoscales on servers
  - Restricted to particular language run-times (standard) with some customization allowed (flex)
- Serverless functions
  - Also implemented with containers
  - Similar operation at function-level

- Serverless containers
  - BYOC!
  - Effectively the same, but you build the container image and ask the cloud to run it as with serverless platforms/functions

Container support services

- Container Registry (`gcr.io`) (e.g. hosted, private DockerHub)
  - Store container images used in your project within GCP for security and quick instantiation
  - Integrated as one of the providers in `docker pull` and `docker run` commands
- Container Builder (e.g. `docker build`)
  - Remotely build container images on GCP

Google Cloud Run

- User supplies container containing entire run-time
  - Cloud Run dynamically instantiates and runs function within it on-demand
  - Replicates container under high-load
  - Removes containers when not being used
  - https://cloud.run
- Can deploy straight from a git repo!
  - https://github.com/GoogleCloudPlatform/cloud-run-button
- See labs

Distributed messaging

Portland State
Computer Science

Message Brokers

- Also known as publish-subscribe messaging systems
- Messaging in the cloud to sending and receive event notifications
  - Used to trigger functions
  - Used as connectors for streaming infinite data streams through data processing pipelines
  - Must be interoperable across multiple languages and platforms to connect heterogeneous producer/consumers of data
  - Must scale

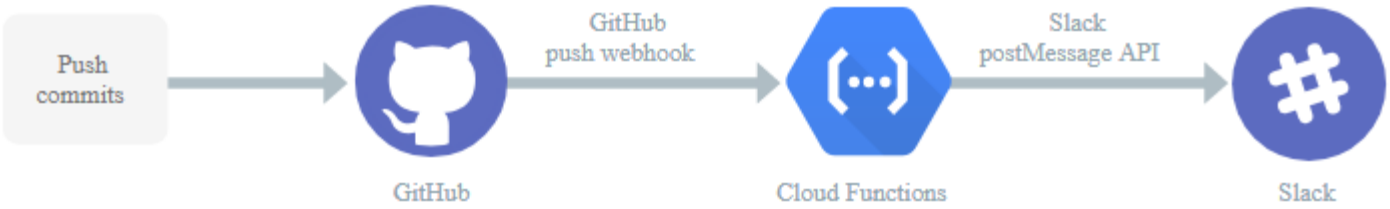
Google Cloud Pub/Sub

- Many-to-many asynchronous messaging in GCP
  - > 1M messages per second
- Used to pipe data into App Engine, BigQuery, Dataflow
- Often used as triggers for Cloud Functions
  - IoT devices and sensors generating data
  - Push notifications for applications

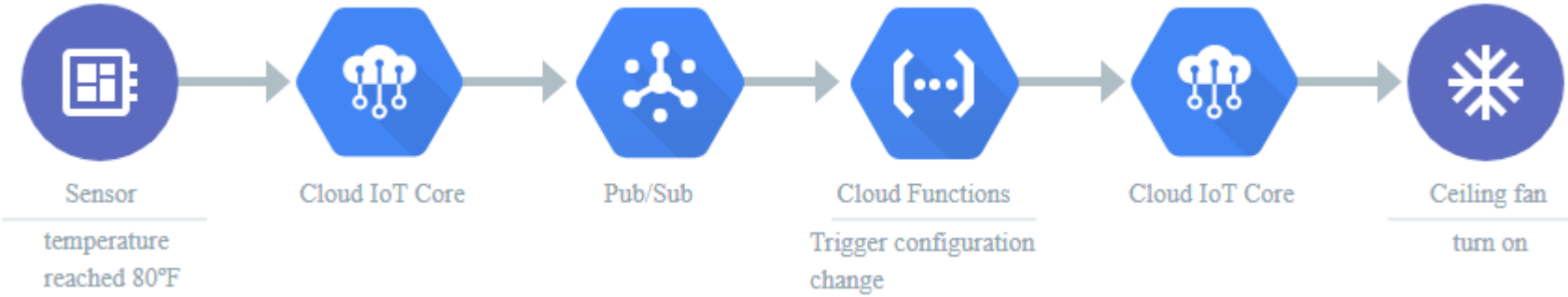
Putting it all together

Portland State
Computer Science

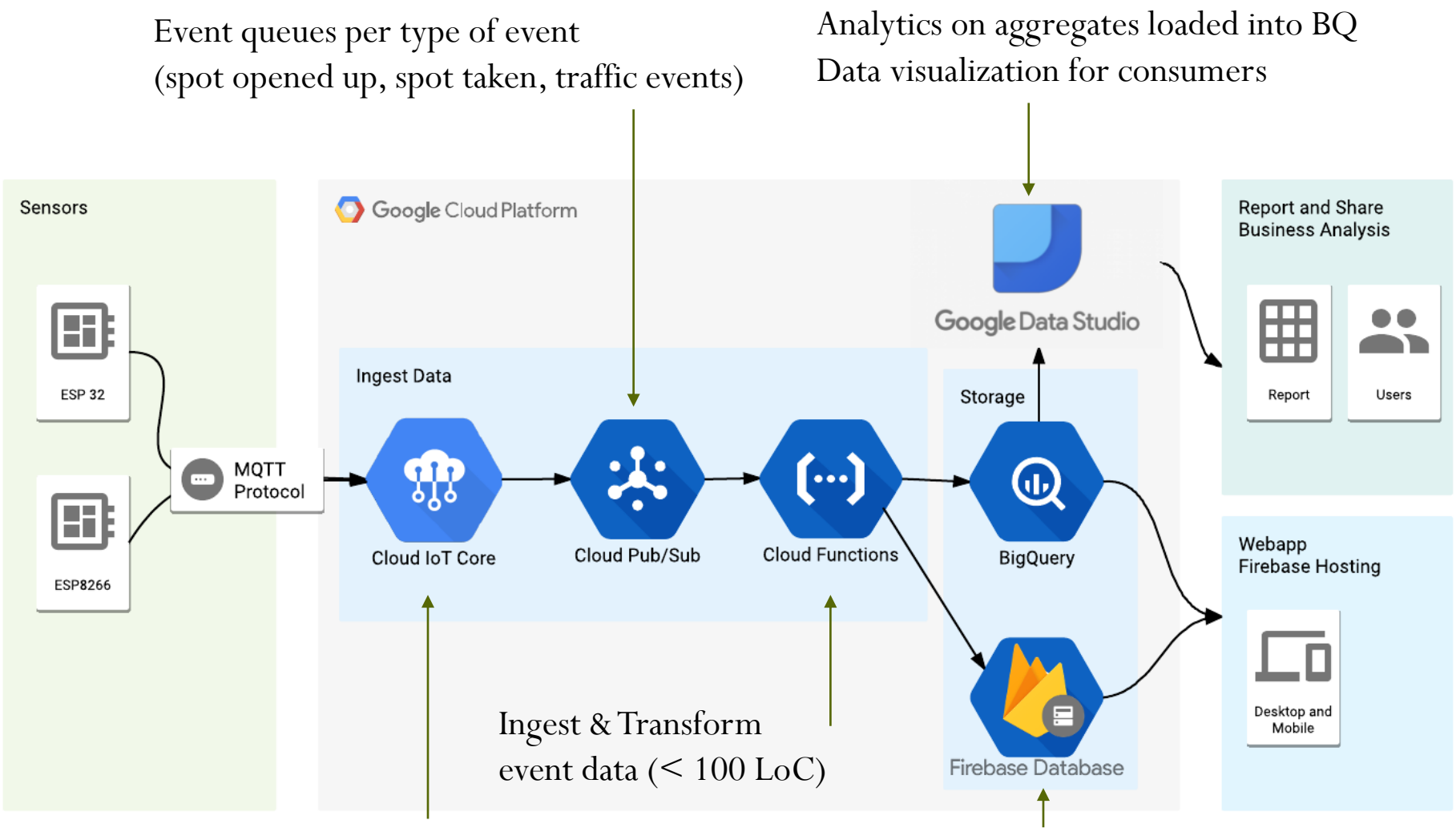
- GitHub integration on our Slack channel



- IoT temperature application



- Smart parking application



- IoT device anomaly detection (akin to senr.io)