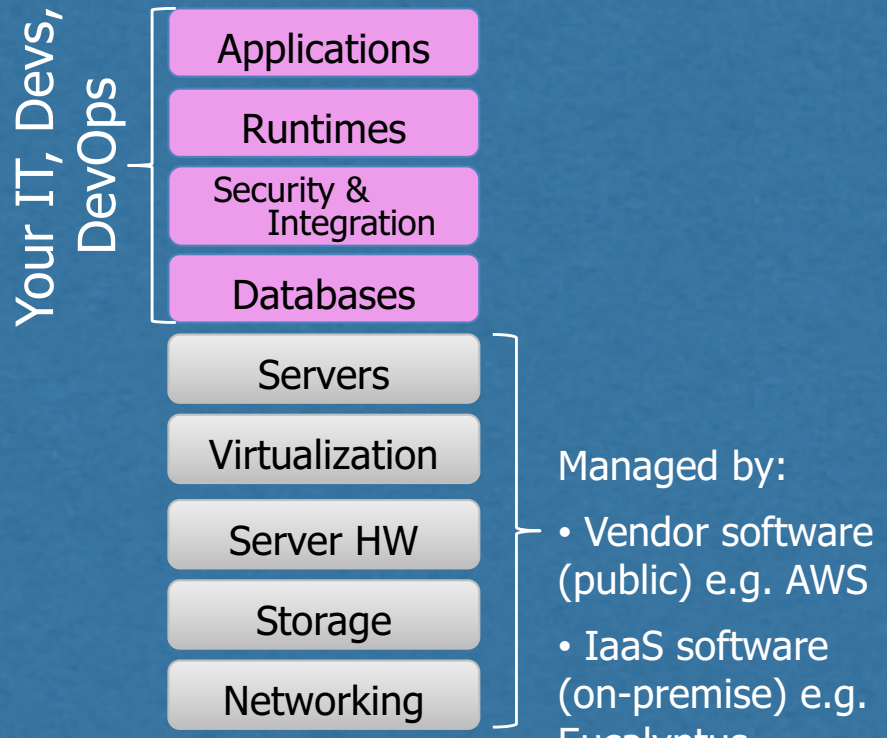
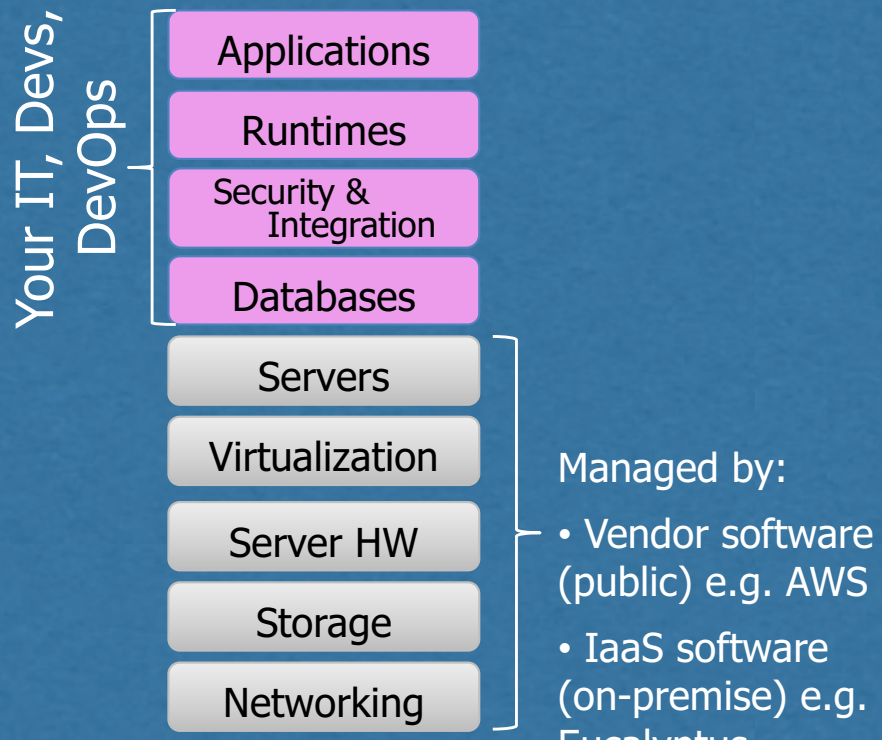
An Introduction to Cloud Platforms-as-a-Service: Google App Engine and AppScale

Chandra Krintz
Professor, CS
UCSB

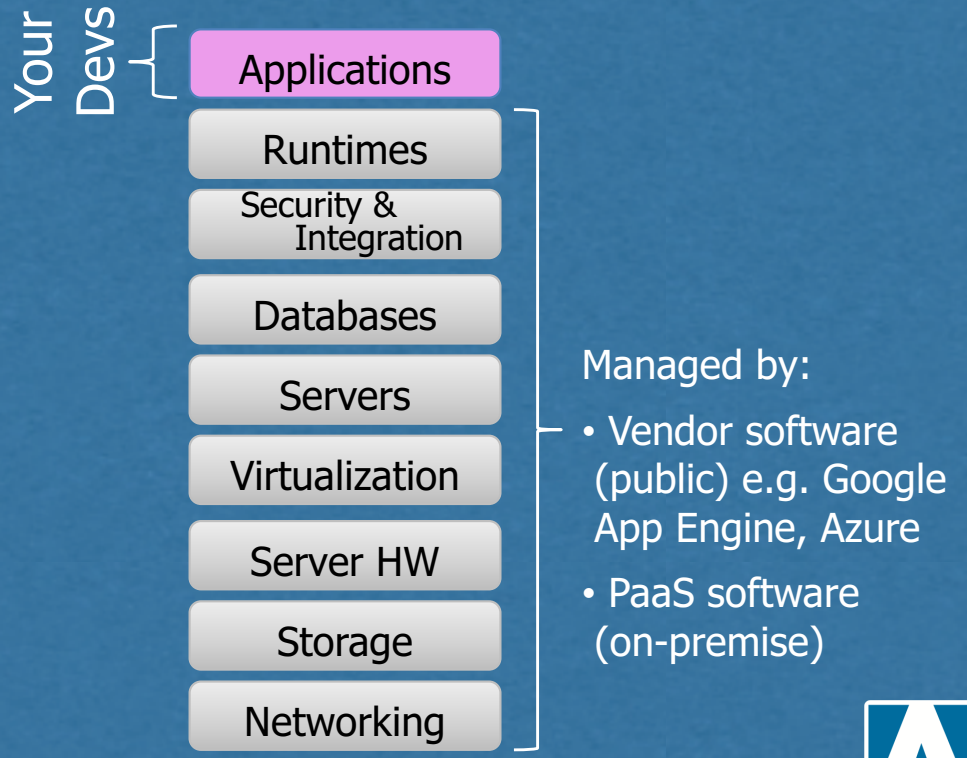
Cloud Infrastructure (as a Service)



Cloud Infrastructure (as a Service)



Cloud Platform (as a Service)



AppScale

Google App Engine

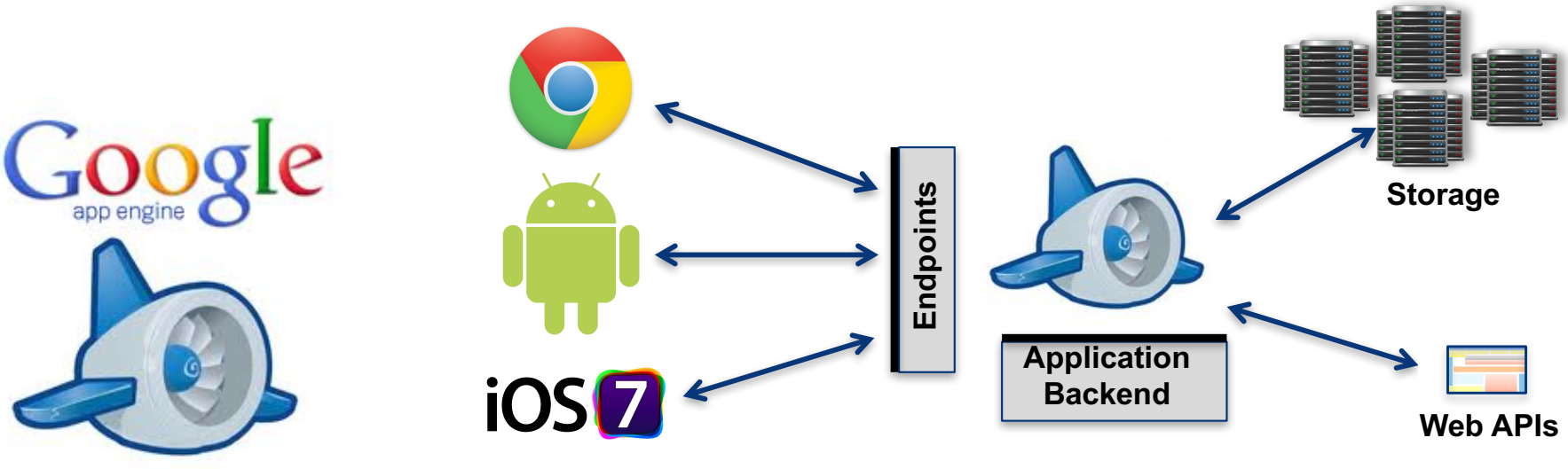
- Google's platform as-a-service – one of the **first** public PaaSs
  - Hosting service for web apps, services, and mobile backends
  - No notion of "server" ***-- yes, serverless before there was serverless***

Google App Engine

- Google's platform as-a-service – one of the **first** public PaaSs
  - Hosting service for web apps, services, and mobile backends
- The result of over a decade of studying Googlers
  - Make them more productive, innovative, and satisfied

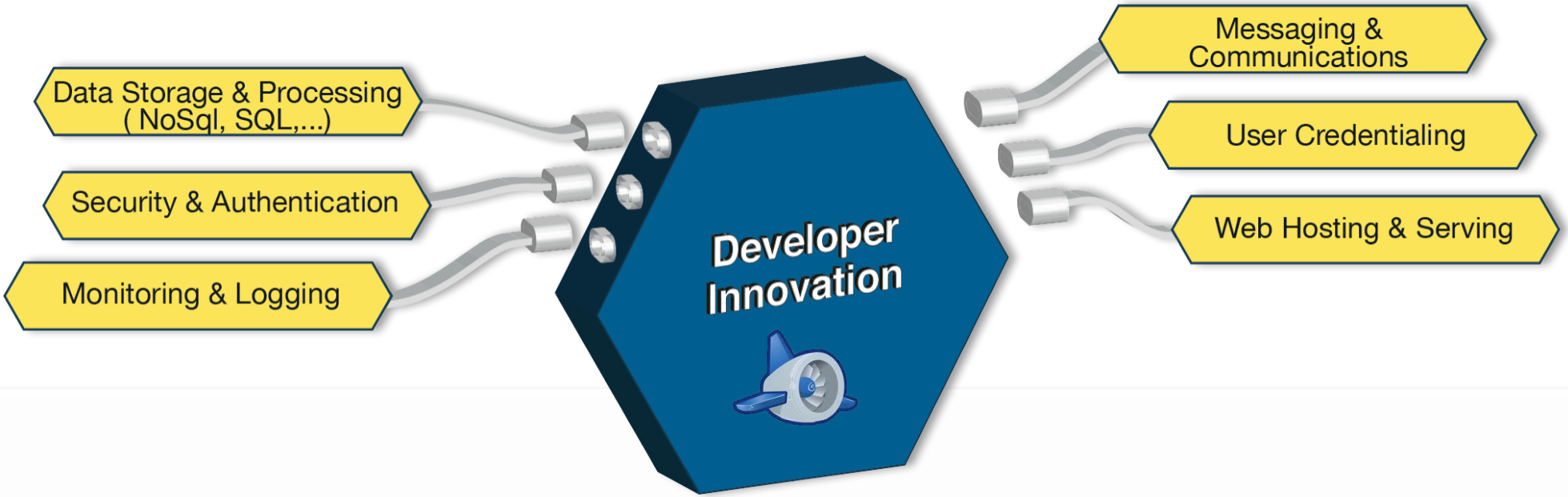
Google App Engine

- Google's platform as-a-service – one of the **first** public PaaSs
  - Hosting service for web apps, services, and mobile backends
- Made publicly available in 2008 (preview) in GA 2011
  - Now over **7 million** active apps, >½ of all Internet accesses/week use it



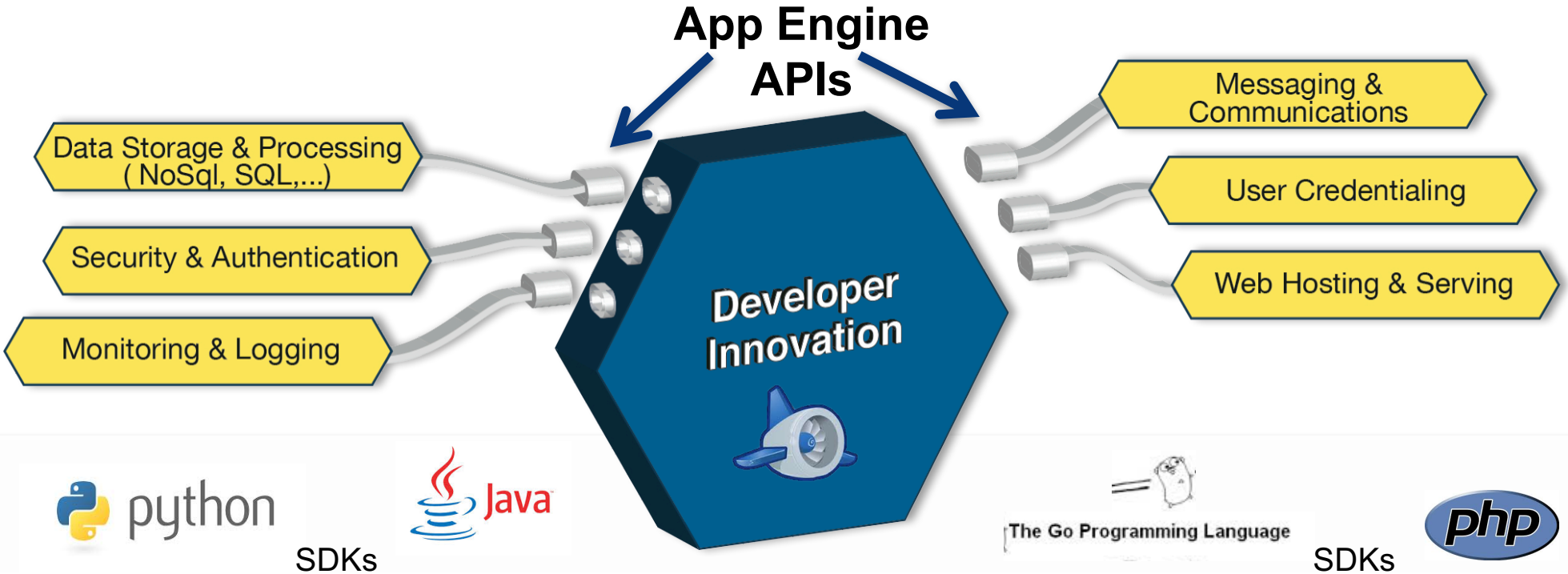
App Engine Development & Deployment

- Decouples app/innovation from common services
  - Share scalable services across apps
  - Automatically manages and scales apps + service ecosystem



App Engine Development & Deployment

- Decouples app/innovation from common services
  - Share scalable services across apps
  - Automatically manages and scales apps + service ecosystem



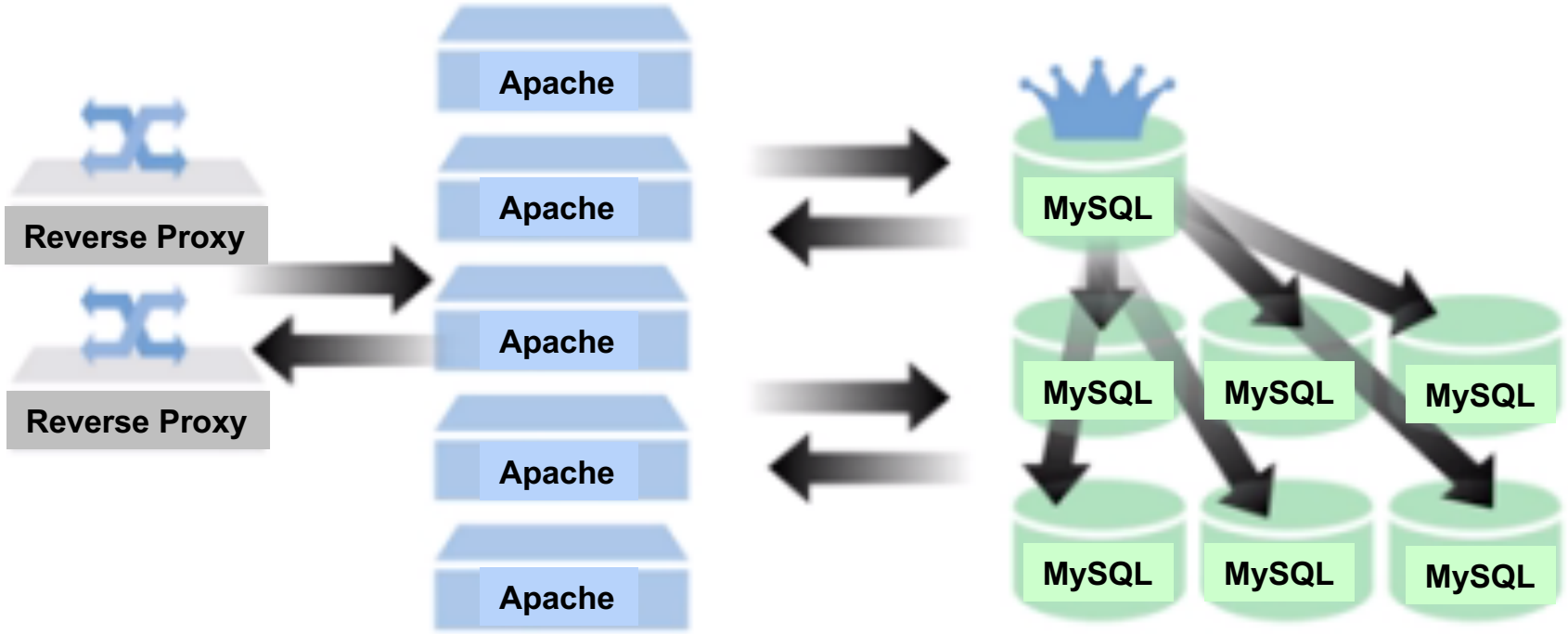
App Engine App & Deployment Model

- Everything is a web request or background task
- Sandboxed execution
  - Restrictions for scale/safety, quotas on free use
    - No file system access
    - Data persistence via Datastore, memcache, and Cloud SQL
    - Processing limits (frontends and tasks)
    - Language libraries limited to “white list”

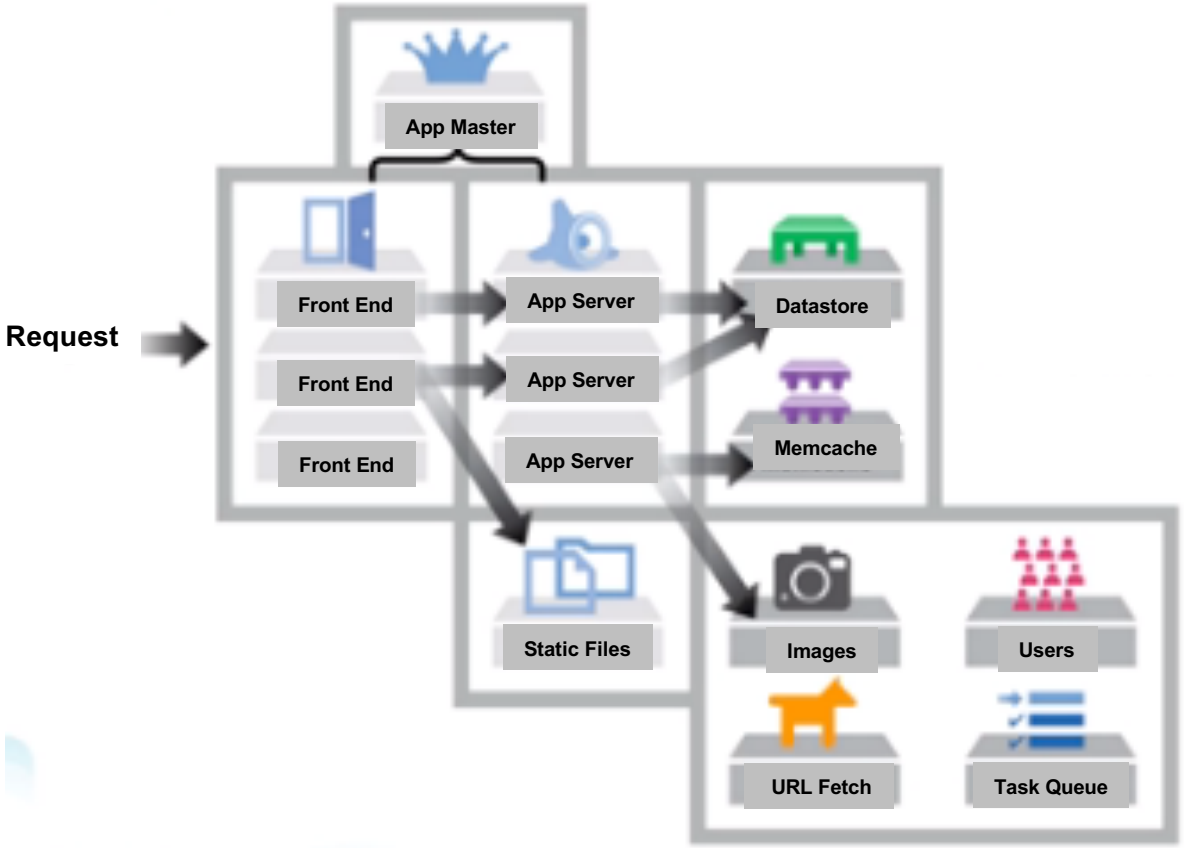
App Engine App & Deployment Model

- Everything is a web request or background task
- Sandboxed execution
  - Restrictions for scale/safety, quotas on free use
    - No file system access
    - Data persistence via Datastore, memcache, and Cloud SQL
    - Processing limits (frontends and tasks)
    - Language libraries limited to “white list”
  - Quotas (free and billed)
    - In/Out bandwidth
    - Datastore usage
    - Other APIs (Mail, messaging, URL Fetch, …)

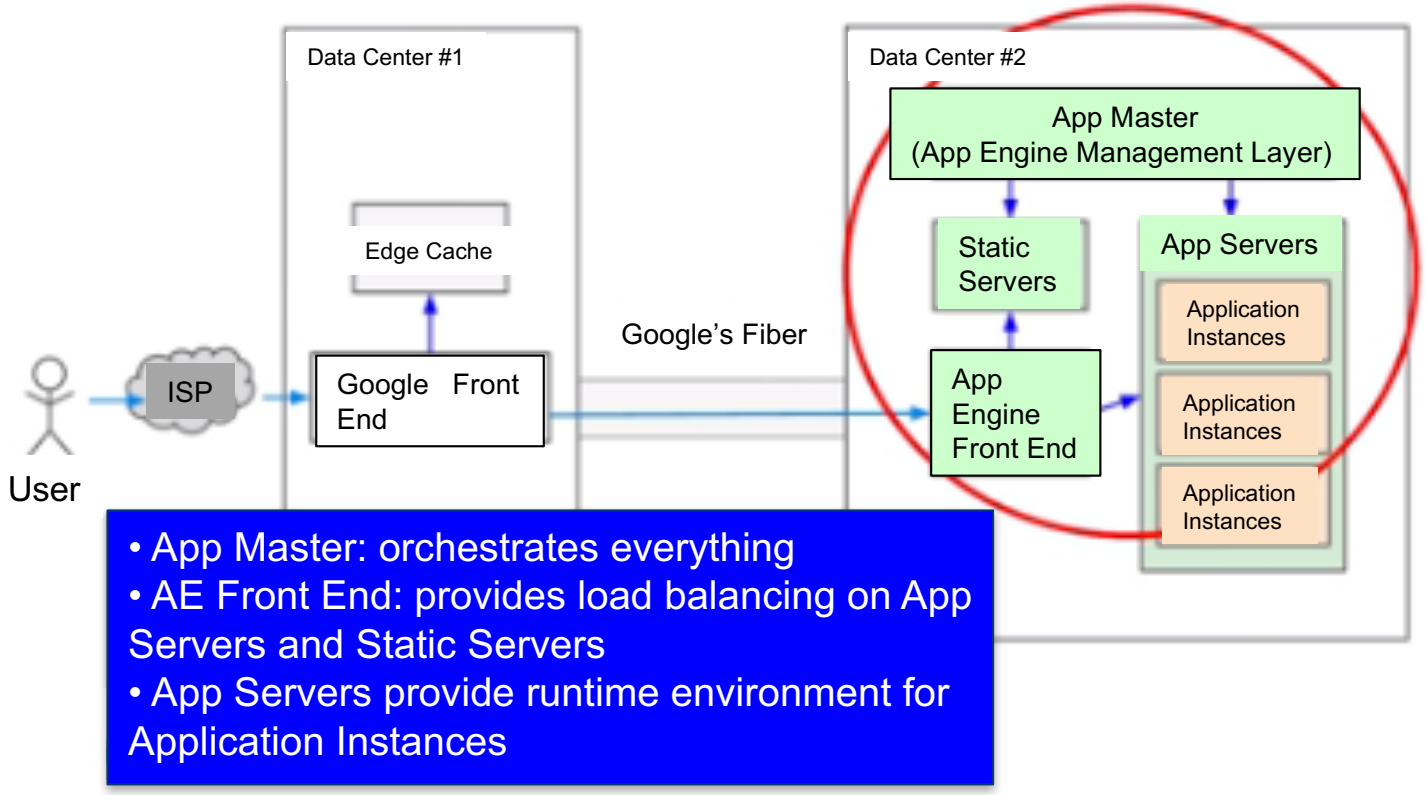
Traditional Website Implementation



App Engine System



App Server System



- App Master: orchestrates everything
- AE Front End: provides load balancing on App Servers and Static Servers
- App Servers provide runtime environment for Application Instances

App Server Application Instances

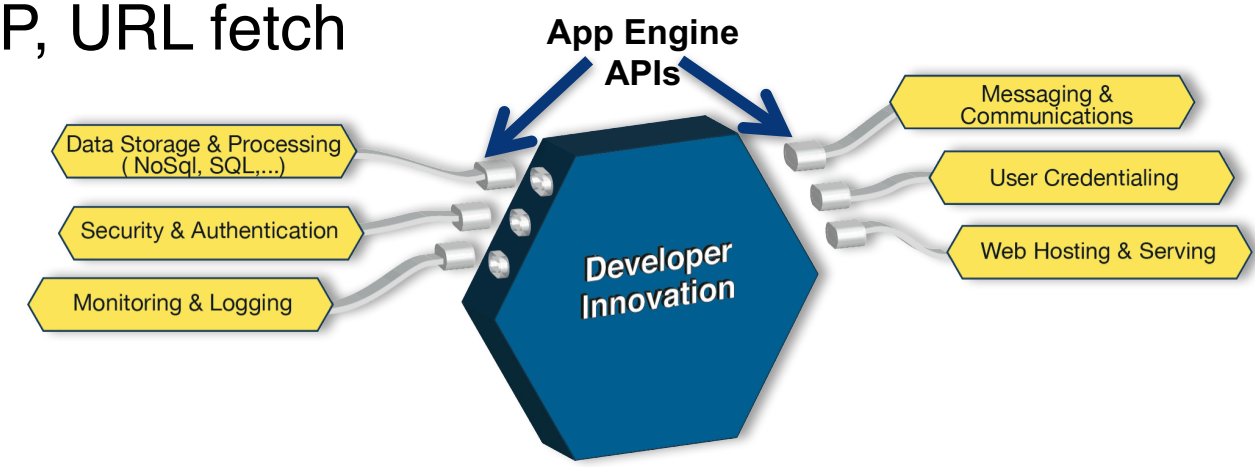
- Frontend instance
  - Is not an App Engine Front End (its an “application instance”)
  - Dynamically created and deleted = low cost
  - Enforce fast response and stateless design
  - Suitable for processing short lived requests
  - 60 sec request limit, 10mins for tasks, 32MB memory payload

App Server Application Instances

- Frontend instance
  - Is not an App Engine Front End (its an “application instance”)
  - Dynamically created and deleted = low cost
  - Enforce fast response and stateless design
  - Suitable for processing short lived requests
  - 60 sec request limit, 10mins for tasks, 128MB memory
- Backend instance (another type of “application instance”)
  - Statically created and deleted = higher cost
  - No limit for response time, supports stateful design
  - Suitable for batch processing
- Both billed on instance hours

App Engine APIs and Implementations (Services)

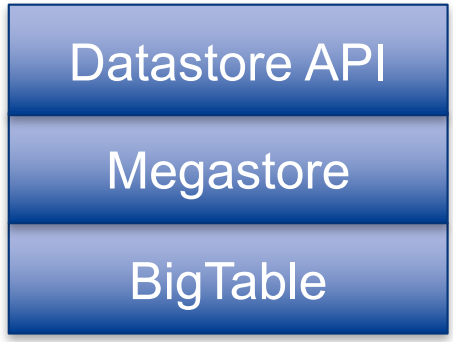
- **Datastore** – key/value object persistent storage
  - Fast, replicated, and scalable for large-scale data
- Memcache – key/value in-memory cache (not persistent)
- Task queue, cron, pipelines, map reduce
- Blobstore: key/value persistent storage for large objects
- Users, mail, search, XMPP, URL fetch



App Engine Datastore API

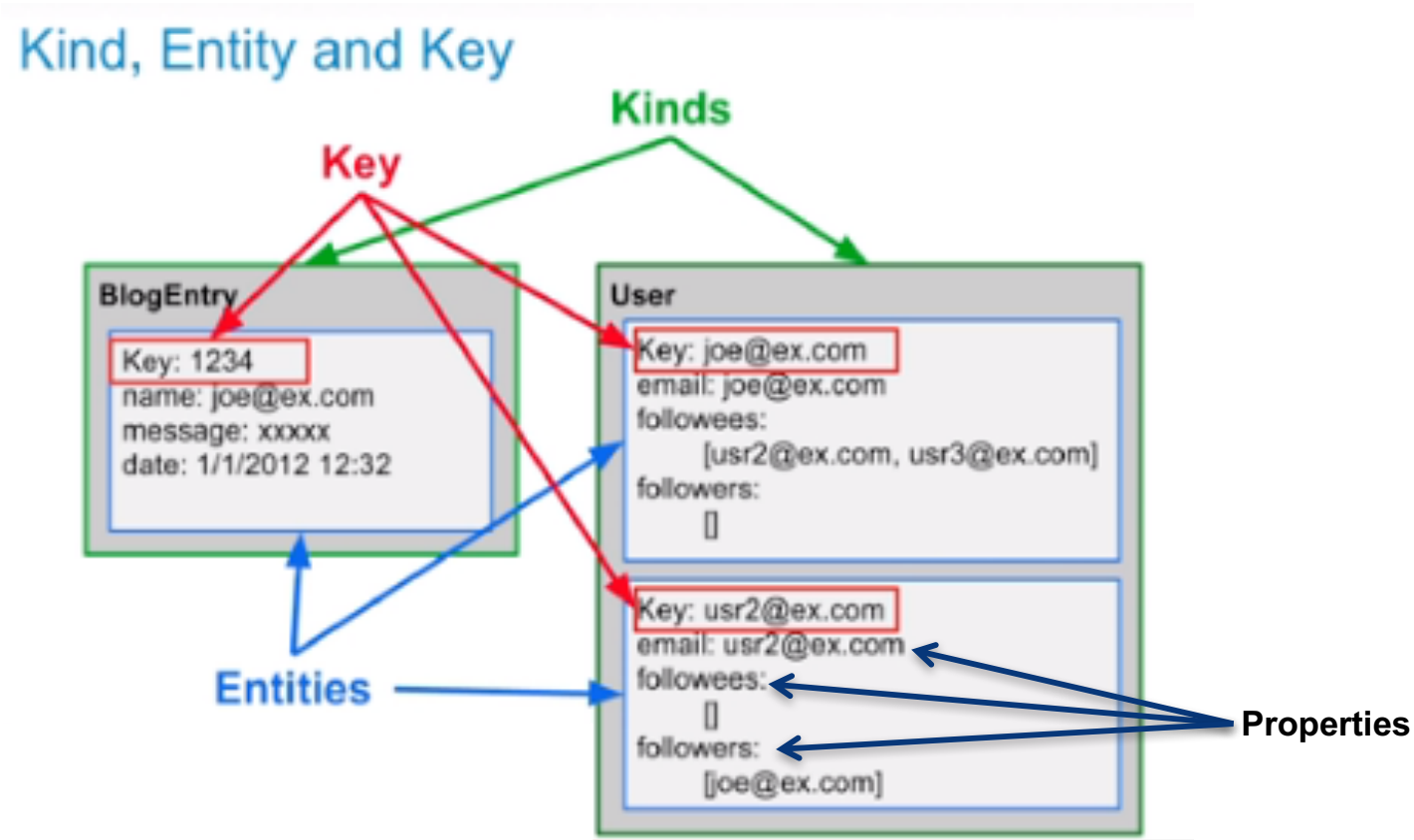
- Datastore – key/value object persistent storage (dynamic schema)
  - Fast, replicated, and scalable for large-scale data
  - Easily extensible entity structures (kinds), less than 1MB in size
  - Simple API: put, get, delete, range_query (subset of SQL)
    - Only simple queries supported, limited indexing support
  - Strongly consistent entity writes (row-level atomic updates)
    - Limited transaction support for multi-entity atomic updates

	Datastore	RDBMS
Category of object	Kind	Table
One entry/object	Entity	Row
Unique identifier of data entry	Key	Primary Key
Individual data	Property	Field



Google Data Center

Kind, Entity and Key



Creating an Entity with Java Low-level API

```
DatastoreService datastore =
    DatastoreServiceFactory.getDatastoreService();

Entity employee = new Entity("Employee");
employee.setProperty("name", "Antonio Salieri");
employee.setProperty("hireDate", new Date());
employee.setProperty("attendedHrTraining", true);
datastore.put(employee);
```

Creating an Entity with Python Low-level API

```
class Employee(db.Model):
    name = db.StringProperty(required=True)
    hire_date = db.DateProperty()
    new_hire_training_completed =
        db.BooleanProperty(indexed=False)

e = Employee(name="Antonio Salieri",
    email=users.get_current_user().email())
e.hire_date = datetime.datetime.now().date()
e.put()
```

Java

- Low level API
  - The best performance, but more coding
- JDO/JPA
  - More portability by Java standard APIs
- Third party frameworks
  - Objectify, Twig, Slim3...
  - Sophisticated features with better performance

Python

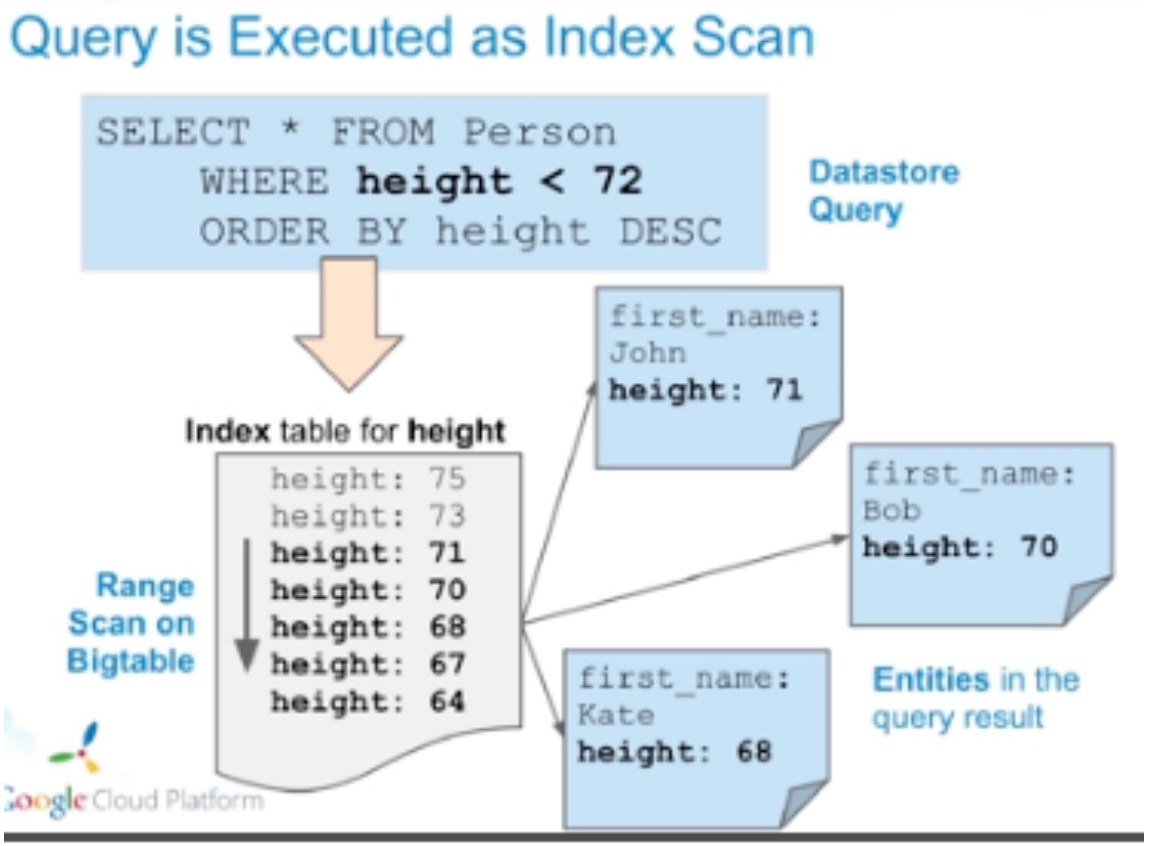
- DB API
  - Traditional Datastore API for Python
- NDB API (New DB)
  - Automatic caching, sophisticated queries, atomic transactions

```
class Person(ndb.Model):
    user = ndb.UserProperty()
    balance = ndb.FloatProperty()
    phone = ndb.StringProperty()
    last_login = ndb.DateTimeProperty()
```

```
class Person(ndb.Model):
    ...

new_person = Person()
new_person.put()
```

DATASTORE QUERIES



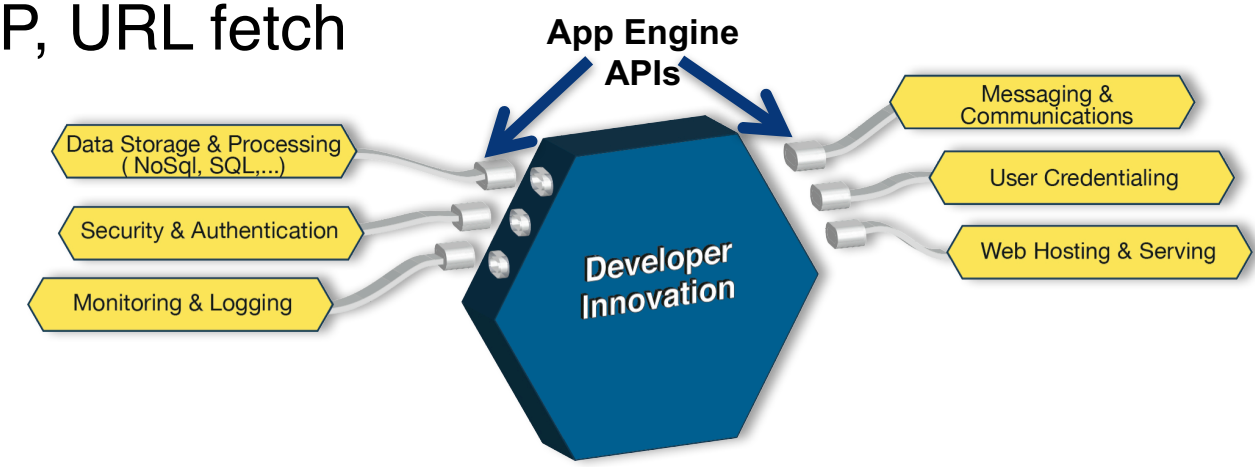
DATASTORE QUERIES

- No joins, aggregate functions, search
- All sorts are performed ahead of time
- Single property (autogen'd index for each column)
- Composite (multi-property) – index must be specified

- Entity groups (related Kinds) – Restricted: 1 update per second
  - Defines the scope of a transaction
  - Hierarchical relationships: Parent->Child->GrandChild
    - ACID transactions within entity groups (optimistic locking)
    - Ancestor queries

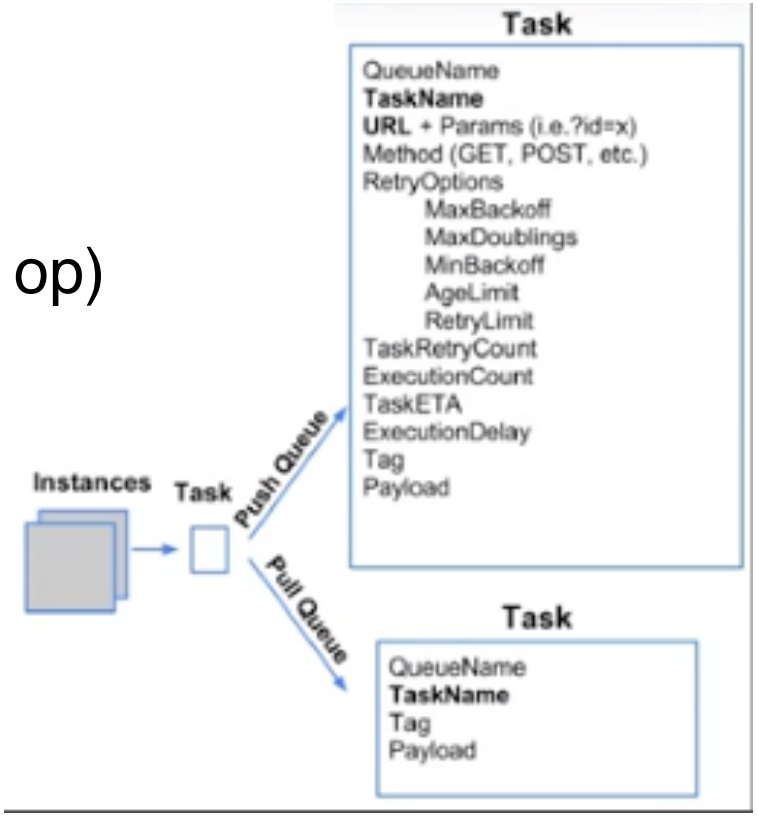
App Engine APIs and Implementations (Services)

- Datastore – key/value object persistent storage
  - Fast, replicated, and scalable for large-scale data
- Memcache – key/value in-memory cache (not persistent)
- **Task queue**, cron, pipelines, map reduce
- Blobstore: key/value persistent storage for large objects
- Users, mail, search, XMPP, URL fetch



App Engine Task Queues

- Task: unit of work
  - Write object to datastore
  - Send an email
- All versions of an application share queues
- Push queue for automatic execution (HTTP op)
  - Fully managed, retry support
- Pull queues for programmatic consumption
- Task have unique names
  - Generated automatically if not assigned
  - Insert new task with same name will fail



App Engine Task Queues

The task queue is a simple way to perform work outside of a user request.

Push Queue:

Features:

- Executed ASAP
  May cause new instances
  Frontend or Backend
  - 10min or unlimited
  - Max 100K task size

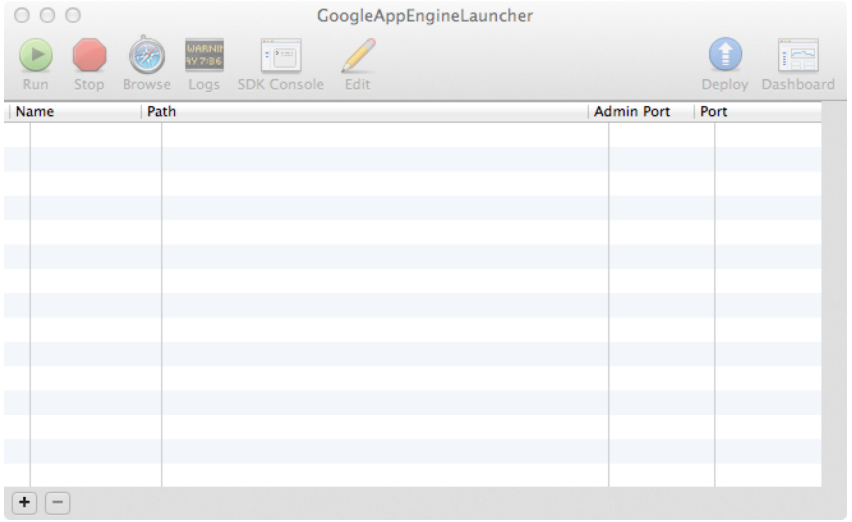
Pull Queue:

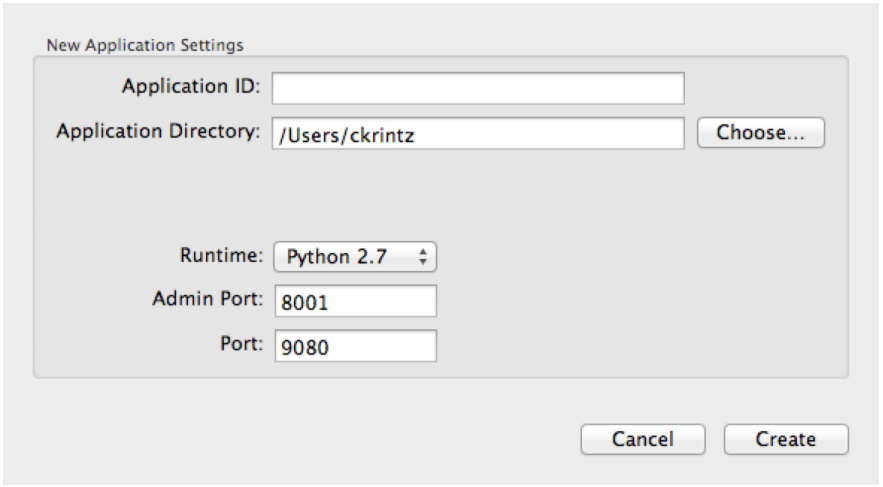
Features:

- Task leased by worker
  REST interface with ACL
  - Consumer can be outside GAE
  - Max 1MB task size

Programming App Engine

- Download the SDK from Google
- Create an app ID via the admin console: https://appengine.google.com/
- Program your app
  - Including **program configuration files**
  - Autogenerated with GoogleAppEngineLauncher (Mac, Windows)





App Configuration Files

- Python app.yaml – in top level app directory
  - # start single line comments
  - POSIX regex syntax
  - Autogenerated by GoogleAppEngineLauncher
  - Run app via dev_appserver.py
    - Indexes autogenerated
    - Stored in file under #AUTOGENERATED

```
application: myapp
version: 1
runtime: python27
api_version: 1
threadsafe: true

handlers:
- url: /
  script: home.app

- url: /index\.html
  script: home.app

- url: /stylesheets
  static_dir: stylesheets

- url: /(.*\.(gif|png|jpg))
  static_files: static/\1
  upload: static/(.*\.(gif|png|jpg))

- url: /admin/.*
  script: admin.app
  login: admin

- url: /.*
  script: not_found.app
```

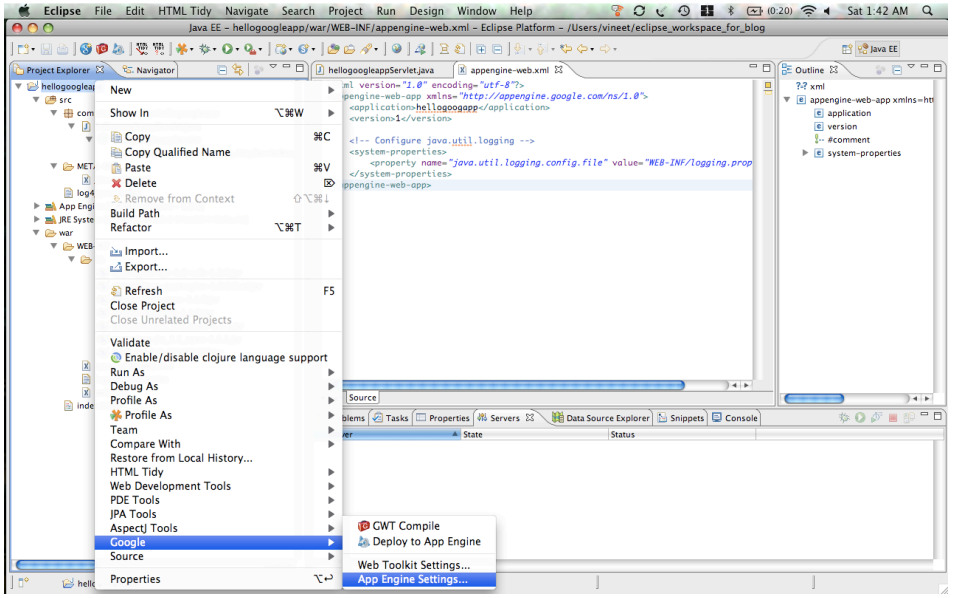
- https://developers.google.com/appengine/docs/[python,java]/config/appconfig

App Configuration Files

- Python app.yaml – in top level app directory
  - # start single line comments
  - POSIX regex syntax
  - Autogenerated by GoogleAppEngineLauncher
  - Run app via dev_appserver.py
    - Indexes autogenerated
    - Stored in file under #AUTOGENERATED
- Java can be done similarly WEB_INF/app.yaml
  - Autogenerates WAR xml files
  - Indexes autogenerated: WEB_INF/index.yaml
- https://developers.google.com/appengine/docs/[python,java]/config/appconfig

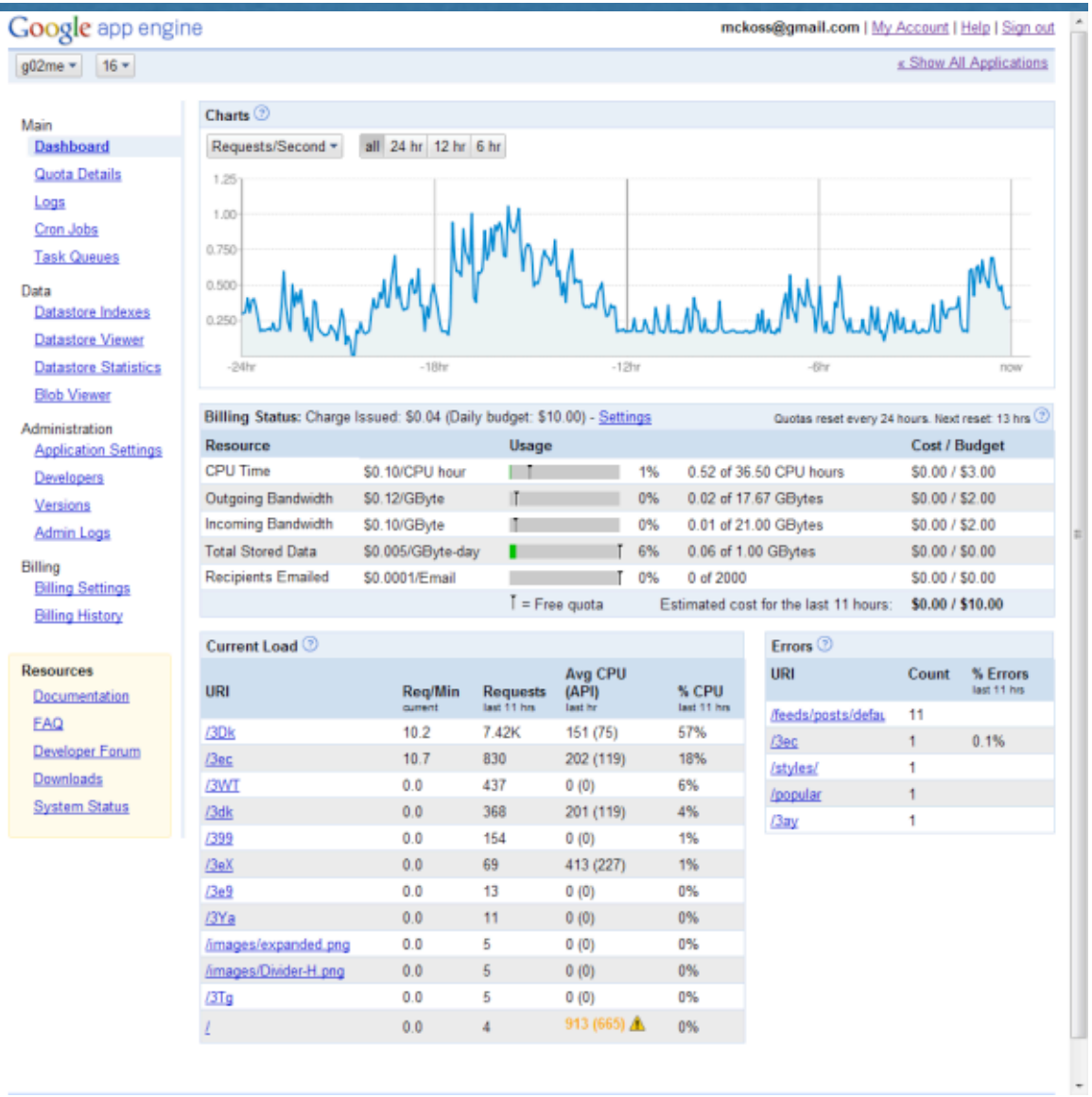
Deploying to App Engine

- Program your app, test with the SDK, generate Datastore indexes
- Update configuration files (fine tune as needed)
- Upload your app to Google's resources for execution
- Maven (mvn) support for CLI automation



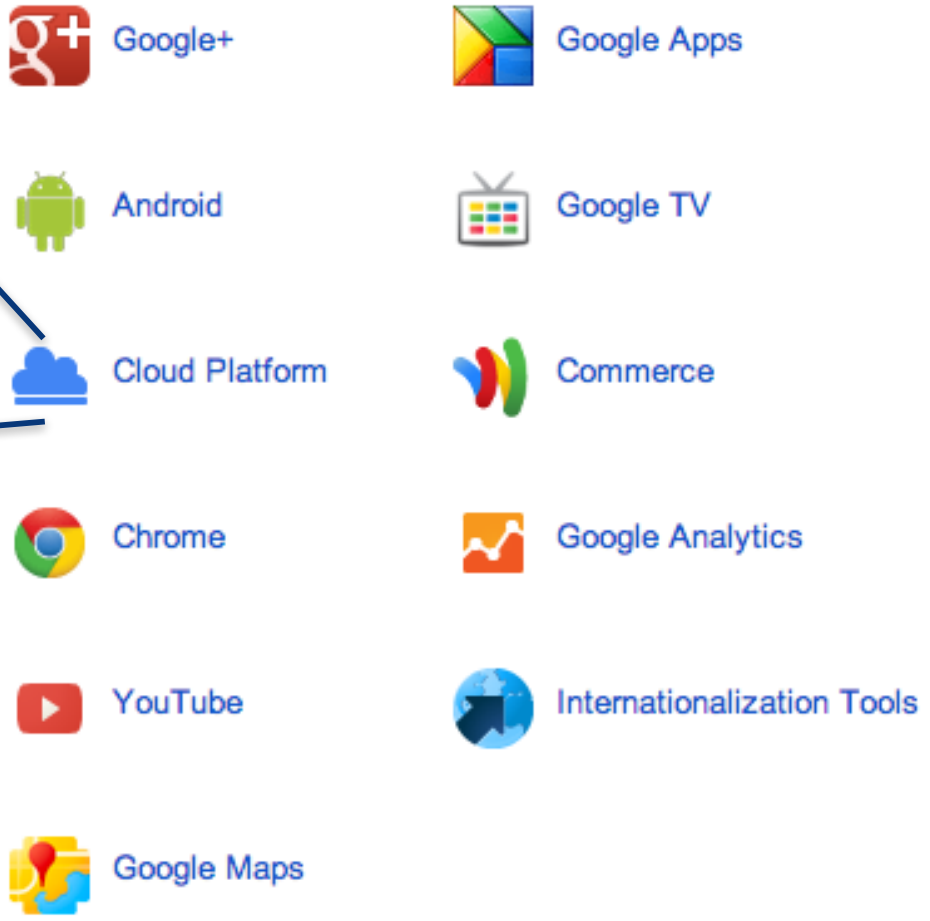
Admin Console

- https://appengine.google.com/
- Usage updated daily
- https://developers.google.com/appengine/docs/adminconsole/



Google Cloud Platform & Technologies

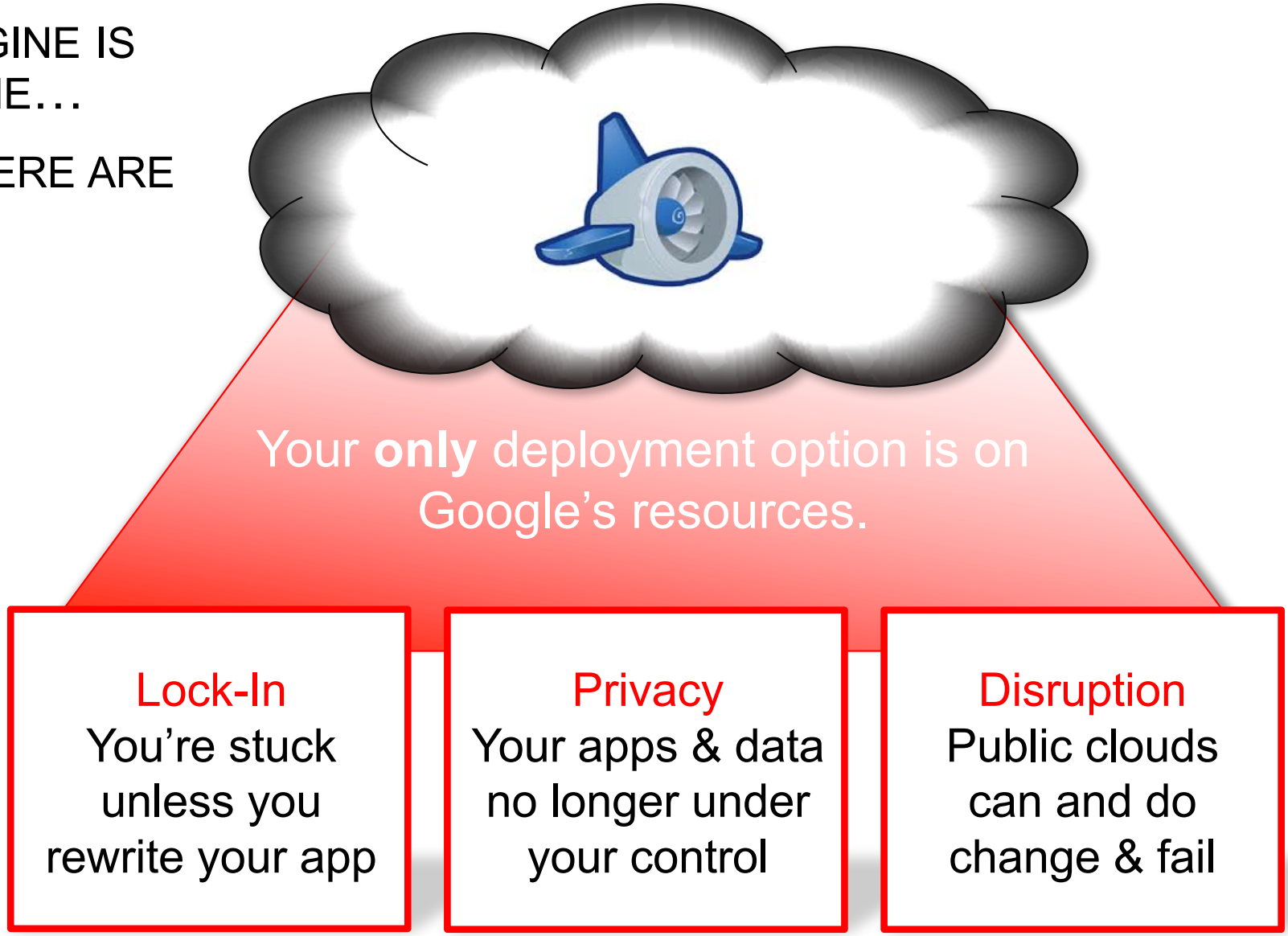
- App Engine (PaaS)
- Compute Engine (IaaS)
- Cloud Storage
- Cloud SQL
- BigQuery



- Technology APIs…
- Linked via service accounts
  - Billing must be enabled
  - https://developers.google.com/accounts/dc
    s/OAuth2#serviceaccount

App Engine is Awesome…

But! There are Cloudy Issues



The Solution: AppScale

- Mirrors Google App Engine
  - Using ***open source*** & other cloud services

AppScale Mirrors Google App Engine

Your App Here

Google
app engine

AppScale Mirrors Google App Engine

AppScale Mirrors Google App Engine

- On-premise
- Behind your firewall
- Everywhere

Google app engine

NO CODE REWRITE

Your App Here

AppScale

Google Compute Engine

openstack™

Windows Azure

amazon web services™

vmware

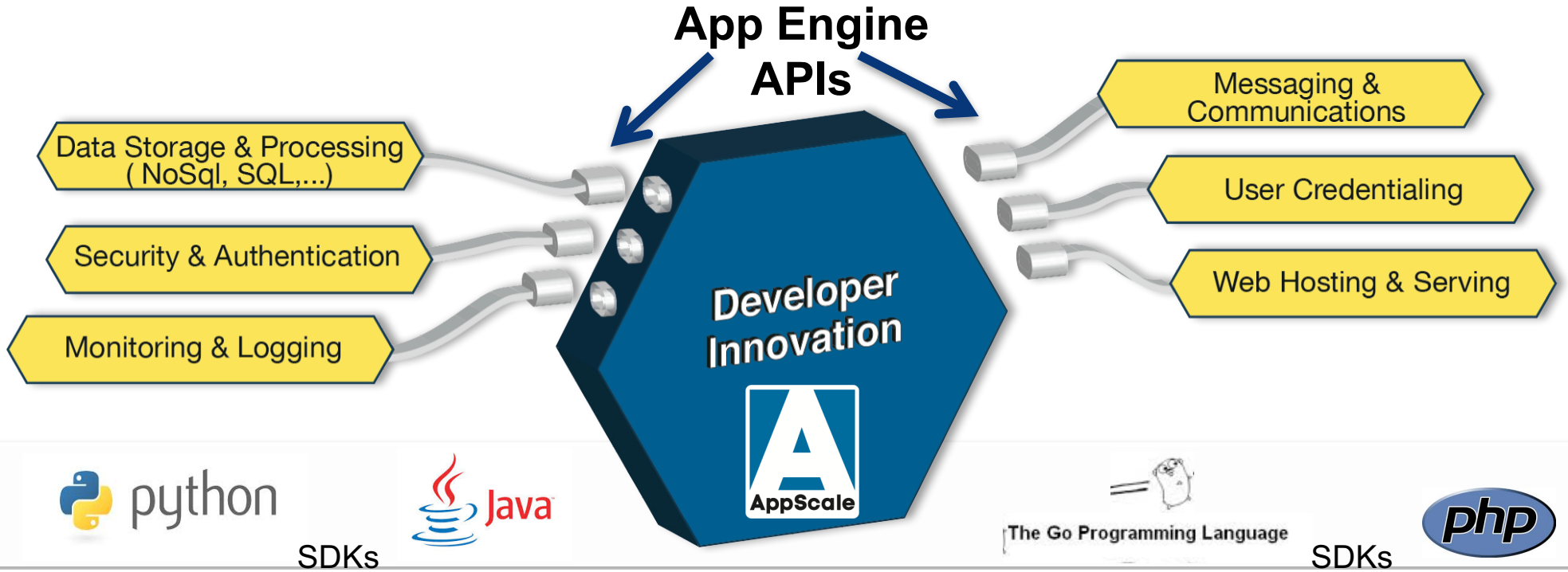
KVM

cloudstack

EUCALYPTUS

THE SOLUTION: APPSCALE

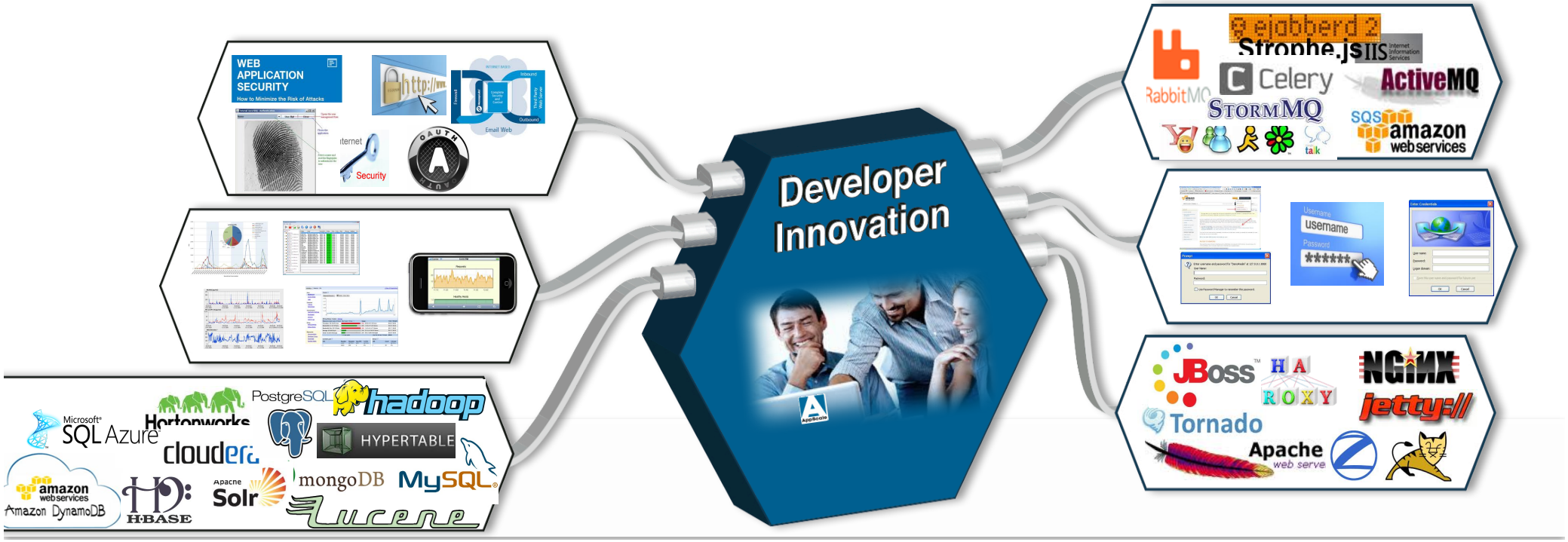
- Mirrors Google App Engine using open source & other cloud services
  - Implements the App Engine APIs
  - Automatically manages and scales apps + service ecosystem



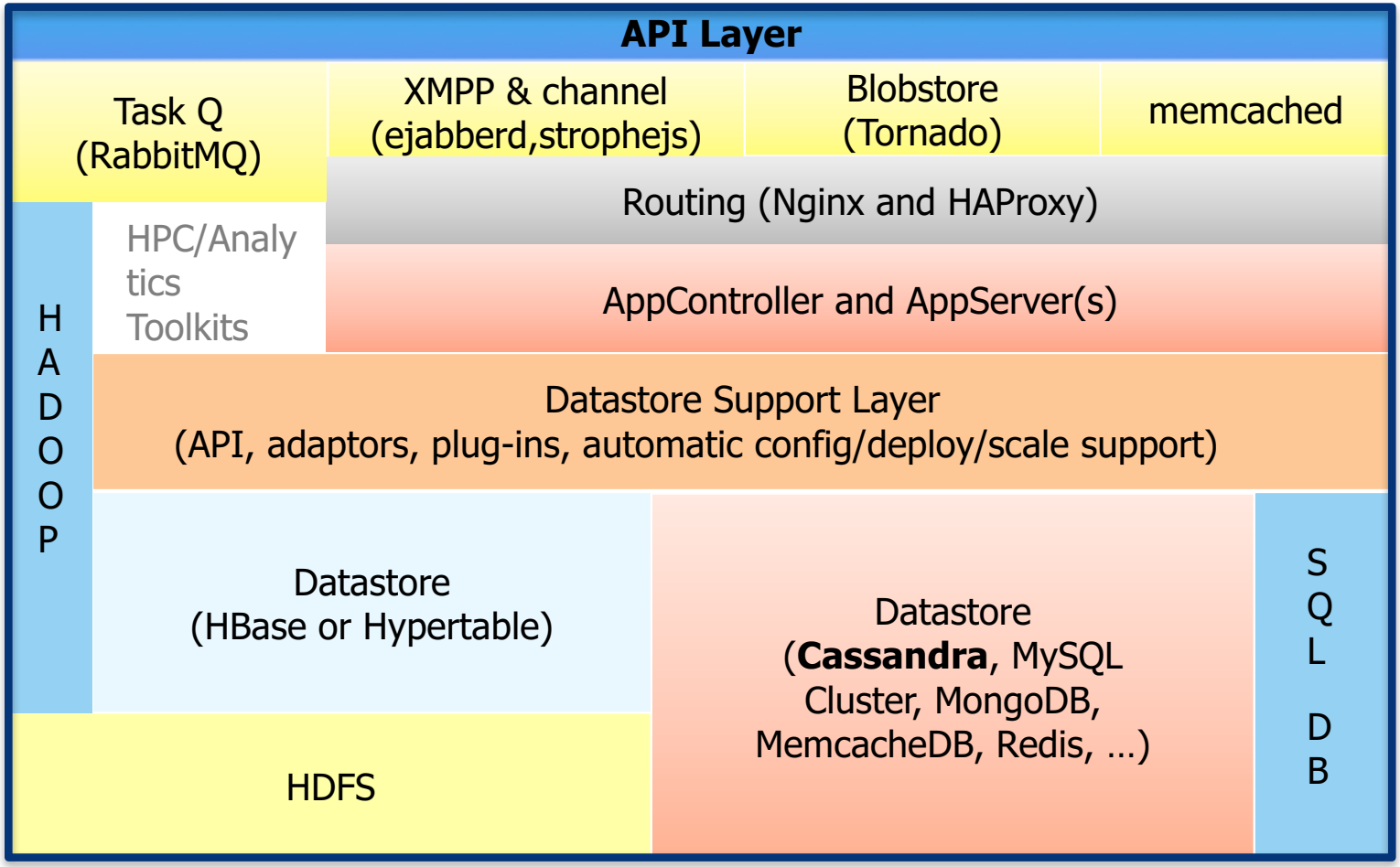
AppScale Extensions & Research

- Plug/play multiple alternatives for each:
  open source, public cloud, proprietary, legacy
- API governance

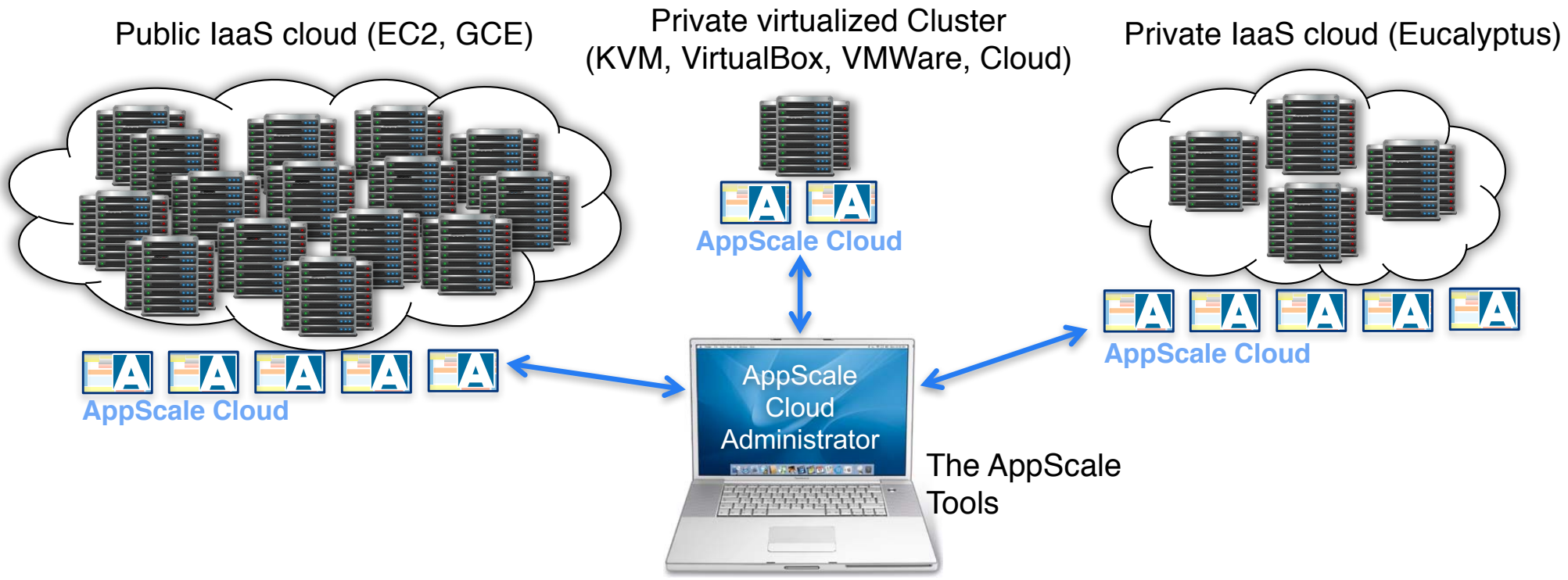
NO CODE REWRITE



AppScale Software Stack

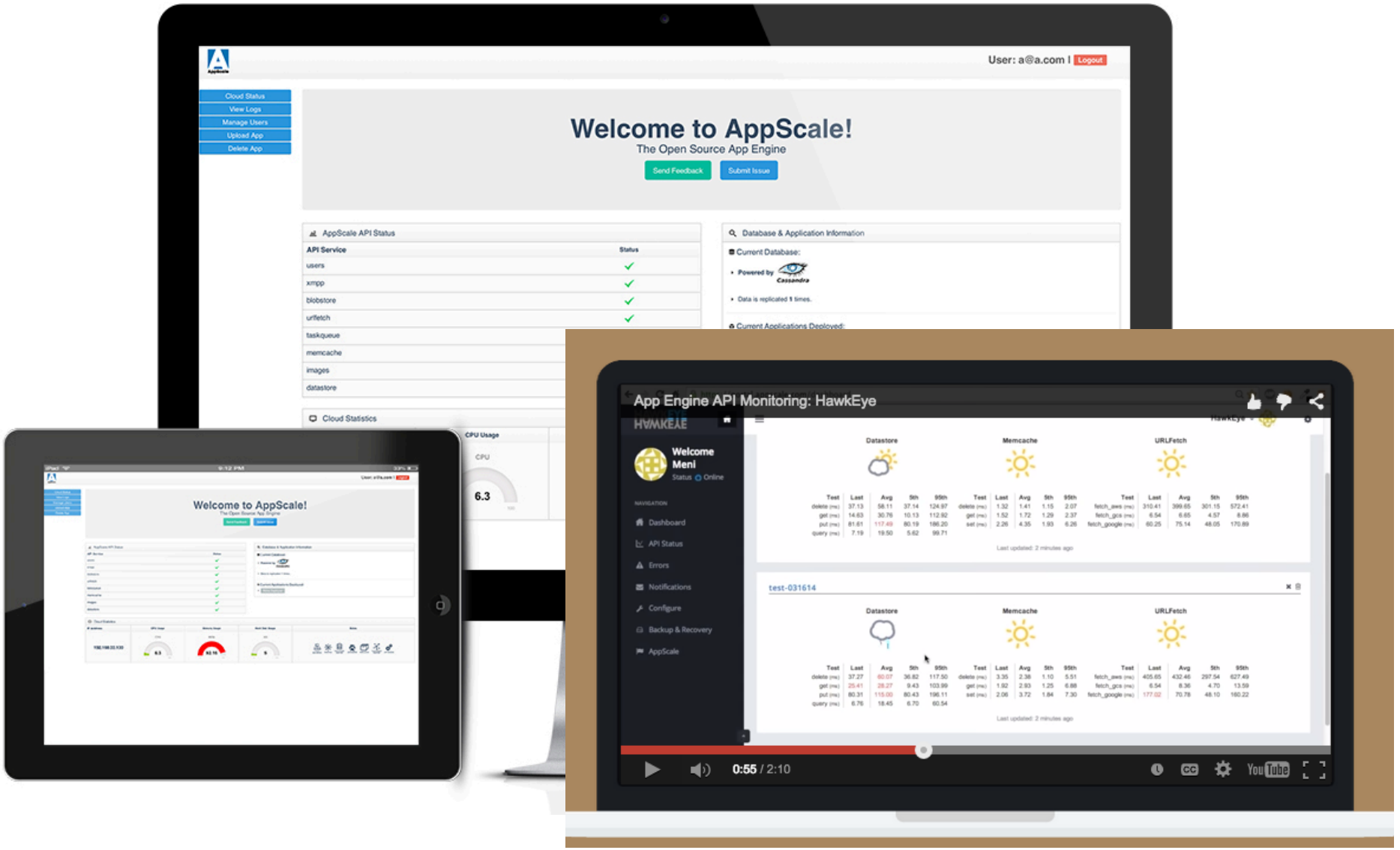


EASY APPSCALE PAAS DEPLOYMENT



AppScale VM image/instance
contains complete SW stack

Each instance takes on 1+ *roles*


User: a@a.com | Logout
Cloud Status
View Logs
Manage Users
Upload App
Delete App
Welcome to AppScale!
The Open Source App Engine
Send Feedback
Submit Issue
AppScale API Status
API Service
Status
users
xmpp
blobstore
urlfetch
taskqueue
memcache
images
datastore
Database & Application Information
Current Database:
Powered by
Cassandra
Data is replicated 1 times.
Current Applications Deployed:
Cloud Statistics
CPU Usage
CPU
6.3
Welcome to AppScale!
App Engine API Monitoring: HawkEye
Welcome
Meni
Status Online
Dashboard
API Status
Errors
Notifications
Configure
Backup & Recovery
AppScale
Datastore
Memcache
URLFetch
test-031614
0:55 / 2:10