Serverless, NoOps and DevOps are All Marketing Bunkus!

Scott Thomson, Customer Solutions & Innovation
DevOps Talks, MEL March 2018
Serverless = No Ops, Right??

✓ Scalable Ops
Shrinking The Scaling Point

- Your Code
- Your Code
- Your Code
- Your Code
- Fns
- PaaS
- Containers
- Virtual Machines
- Dedicated Physical Servers
Agenda

- Introduction
- Planning process and Trade-offs
- High Level Architecture
- Architecture deep dives
  - Serving infrastructure
  - Analytics and Recommendation Engine
  - Supporting tooling
Introduction

Case Study: ProductNow
Company profile

- Existing online retailer
- Large product catalog and high volume of transactions
- Looking to move to a new way to sell products
Initiative Priorities

- Minimize operational complexity
- Prioritize time-to-market over cost optimization
- Continuous Delivery
- Leverage existing product catalog and data where possible
ProductNow Bot

● Mobile and Chat first shopping experience
● Machine learning algorithms for recommending products and guiding users
● Conversational
● Globally available
Product Team

- ~20 devs assigned to work towards a v1.0
- Polyglot developers pulled from existing teams
  - Node.js, Go, Java, Python
- Machine learning embedded in team
- DevOps team members to support software delivery and reliability
Key Decisions
Application Architecture

What the hell have you built.

- Did you just pick things at random?
- Why is Redis talking to MongoDB?
- Why do you even use MongoDB?

Goddamnit  Nevermind
Conway’s law

"organizations which design systems ... are constrained to produce designs which are copies of the communication structures of these organizations."

— M. Conway[ref]
Conway’s law

new system:  

organization:
ProductNow Org Structure

VP Eng or CTO

Web Lead
- Serving
- Chatbot

Data Lead
- ML
- Analytics

DevOps
- Tooling
- Monitoring & Logging
High Level System Architecture

Chat Bot  Web  Mobile

Serving Infrastructure

Data Processing

Analytics  Recommendation Engine

Tooling

CI/CD  Monitoring  Logging
## Choosing Technologies

<table>
<thead>
<tr>
<th>Serving</th>
<th>Data Stores</th>
<th>Data Processing</th>
<th>Tooling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute Engine</td>
<td>Bigtable</td>
<td>BigQuery</td>
<td>Spinnaker</td>
</tr>
<tr>
<td>Kubernetes Engine</td>
<td>Datastore</td>
<td>Pub/Sub</td>
<td>Jenkins</td>
</tr>
<tr>
<td>App Engine</td>
<td>Cloud Storage</td>
<td>Dataproc</td>
<td>Docker</td>
</tr>
<tr>
<td>Firebase</td>
<td>CloudSQL</td>
<td>Dataflow</td>
<td>Stackdriver</td>
</tr>
<tr>
<td>Cloud Functions</td>
<td>Spanner</td>
<td>ML Engine</td>
<td>Terraform</td>
</tr>
<tr>
<td></td>
<td>Firebase RTDB</td>
<td>Genomics</td>
<td>Deployment Manager</td>
</tr>
<tr>
<td></td>
<td>Firestore</td>
<td></td>
<td>Chef/Puppet/Ansible</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Forseti</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Vault</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CloudKMS</td>
</tr>
</tbody>
</table>
Serving Infrastructure
AI powered Chat

Chat Bot | Web | Mobile
---|---|---
Serving Infrastructure

Data Processing
Analytics | Recommendation Engine

Tooling
CI/CD | Monitoring | Logging

Google Cloud
## Serving web content trade-offs

<table>
<thead>
<tr>
<th>App Engine</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>Works for *.googleplex.com</td>
<td>Lots of infrastructure that is not always well hidden Opinionated</td>
</tr>
<tr>
<td>Flex</td>
<td>Simpler to manage Less opinionated</td>
<td>You're running VMs Does not scale to 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>App Engine</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>Serverless Scales to zero Opinionated Javascript-only Express.js is not easy</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Firebase Hosting</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>firebase deploy</td>
<td>Free HTTPS Global CDN zero-cost</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cloud Functions</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Only for static content</td>
</tr>
</tbody>
</table>
# Serving web content trade-offs

<table>
<thead>
<tr>
<th>App Engine Standard</th>
<th>App Engine Flex</th>
<th>Cloud Functions</th>
<th>Firebase Hosting</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pros</strong></td>
<td><strong>Pros</strong></td>
<td><strong>Pros</strong></td>
<td><strong>Pros</strong></td>
</tr>
<tr>
<td>Works for *.googleplex.com</td>
<td>Simpler to manage</td>
<td>Serverless</td>
<td>firebase deploy</td>
</tr>
<tr>
<td>Less opinionated</td>
<td></td>
<td>Scales to zero</td>
<td>Free HTTPS</td>
</tr>
<tr>
<td><strong>Cons</strong></td>
<td><strong>Cons</strong></td>
<td><strong>Cons</strong></td>
<td><strong>Cons</strong></td>
</tr>
<tr>
<td>Lots of infrastructure that is not always well hidden</td>
<td>You're running VMs</td>
<td>Opinionated</td>
<td>Only for static content</td>
</tr>
<tr>
<td>Opinionated</td>
<td>Does not scale to 0</td>
<td>Javascript-only</td>
<td></td>
</tr>
</tbody>
</table>

**Recommendation:** Firebase Hosting
Varying content
If we have a static site, how do we show users their own data?
Exactly like we do on an iOS + Android application, client-side rendering.

● Rich client interactions without reloading the page
● A bit slower to load initially but faster from there
● Works with statically hosted content
● Not great for search engines
Client-side interaction trade-offs

**Traditional API**

**Pros**
- It can be whatever you want

**Cons**
- You have to scale it, secure it, make it fancy

**Pros**
- Works out of the box

**Cons**
- Scalability is limited

**Cloud Firestore**

**Pros**
- Works out of the box

**Cons**
- It's in beta

**Pros**
- Scales horizontally

**Cons**
- You only write app code
Client-side interaction trade-offs

<table>
<thead>
<tr>
<th>Traditional API</th>
<th>Firebase Realtime Database</th>
<th>Cloud Firestore</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pros</strong></td>
<td><strong>Pros</strong></td>
<td><strong>Pros</strong></td>
</tr>
<tr>
<td>It can be whatever you want</td>
<td>Works out of the box</td>
<td>Works out of the box</td>
</tr>
<tr>
<td><strong>Cons</strong></td>
<td><strong>Cons</strong></td>
<td><strong>Cons</strong></td>
</tr>
<tr>
<td>You have to scale it, secure it, make it fancy</td>
<td>You only write app code</td>
<td>Scales horizontally</td>
</tr>
</tbody>
</table>

Recommendation: Cloud Firestore
Cross-platform SDKs

Firebase SDKs are out of the box for iOS, Android, and many more platforms.

All of these work with your existing Firestore back-end. You don't need to build client-side API infrastructure, you just build your app + let Firebase take care of auth, retries, encoding, and all the complexity.

Now our iOS and web codebases, while in different languages look very similar.
Real-time
Effortless offline persistence
What about our bot...
It’s just another client!

- Listen to Firestore in the same manner?
  - Firebase Admin SDK is available for “server-side” use

- Mediate with pub/sub and Cloud Functions
  - SDKs have different capabilities based on language
  - Cloud Functions can listen with “wildcards” in their path
Progressive Web Application
Web Applications which deliver an app-like experience to their end-users.

- Offline-enabled
- Responsive
- Re-engageable
- Installable
- Linkable
Firebase Hosting + Cloud Firestore

- Static content, serve with Firebase Hosting
  - Load data client-side, just like native apps
  - Firebase SDKs give us a lot for “free”

- Firebase has server (admin) SDKs for clients like the chatbot
  - The chatbot is just “another client”, depending on language choice and data-structure pub/sub might be best choice

- Cloud Function triggers + pub/sub to hook up to rest of infrastructure
Data Processing
Data Processing Architecture

Data Processing
- Analytics
- Recommendation Engine

Batch

Streaming
Batch

- **Product Catalog Listing Ingest**: Input from on-prem data warehouse, output to cloud database.
- **Product Catalog Image Ingest**: Input from on-prem data warehouse, output to cloud object store.
- **Recommendation Engine Training**: Input from listing data, and analytics data, output trained model.
Microbatch and Sync

High-level architecture

On-Prem Systems
- Compute
- Storage

Google Cloud Platform

Data drop off → Processing → Online Database

Google Cloud
Solution Trade-offs

Technologies for data drop off

GCE Filer
- Pros:
  - POSIX Compatible
  - Integrates with many processing solutions
- Cons:
  - Self-managed
  - Scaling complexity
  - Always-on and billed

Cloud SQL
- Pros:
  - Ideal for mirroring compatible SQL engines.
  - Transactional
  - Hosted service
- Cons:
  - Not ideal for data drop off, data management overhead.
  - Scaling complexity
  - Always-on and billed

Cloud Storage
- Pros:
  - Serverless, pay as you go
  - High throughput
  - Life cycle policies
- Cons:
  - Object store not POSIX
  - Object write consistency
Solution Trade-offs

Technologies for data drop off

GCE Filer
- Pros: POSIX Compatible
  - Integrates with many processing solutions
- Cons:
  - Self-managed
  - Scaling complexity
  - Always-on and billed

Cloud SQL
- Pros:
  - Ideal for mirroring compatible SQL engines.
  - Transactional
  - Hosted service
- Cons:
  - Not ideal for data drop off, data management overhead.
  - Scaling complexity
  - Always-on and billed

Cloud Storage
- Pros:
  - Serverless, pay as you go
  - High throughput
  - Life cycle policies
- Cons:
  - Object store not POSIX
  - Object write consistency
Solution Trade-offs

Technologies for Data Processing

Hadoop on GCE
- Pros:
  - Easy to move existing pipelines and ETL workloads.
  - Integrates with many data sources
- Cons:
  - Self-managed
  - Scaling complexity
  - Always-on and billed

Cloud Dataflow
- Pros:
  - Managed processing pipelines and auto-scaling.
  - Open source Apache Beam for Batch and Streaming processing models
  - Can run locally, hosted or on Spark.
- Cons:
  - New programming model if not familiar with Beam.
  - Only supports Java and Python SDKs

Cloud Dataproc
- Pros:
  - Fully managed Apache Spark and Hadoop.
  - Can run dataflow pipelines.
  - Integrates with many data sources.
- Cons:
  - Complex streaming model.
Solution Trade-offs

Technologies for Data Processing

Hadoop on GCE
Pros:
- Easy to move existing pipelines and ETL workloads.
- Integrates with many data sources
Cons:
- Self-managed
- Scaling complexity
- Always-on and billed

Cloud Dataflow
Pros:
- Managed processing pipelines and auto-scaling.
- Open source Apache Beam for Batch and Streaming processing models
- Can run locally, hosted or on Spark.
Cons:
- New programming model if not familiar with Beam.
- Only supports Java and Python SDKs

Cloud Dataproc
Pros:
- Fully managed Apache Spark and Hadoop.
- Can run dataflow pipelines.
- Integrates with many data sources.
Cons:
- Complex streaming model.
Streaming Data Processing

Clickstream Data

Input from application analytics telemetry

Output to clickstream database
Solution Trade-offs

Technologies for streaming data

Data Ingestion

Kafka on GCE
- Pros:
  - Simple Producer Interface
  - High throughput
  - Ordering within partitions
- Cons:
  - No REST API
  - Self-managed
  - Scaling complexity

Cloud Pub/Sub
- Pros:
  - Managed global service
  - REST API
  - Pay as you go
- Cons:
  - No ordering of events

Data Processing

Cloud Dataflow
- Pros:
  - Handles unordered data
  - Managed and scalable
  - Integrations with Pub/Sub and Kafka
- Cons:
  - New model for traditional Spark Streaming devs.

Cloud Dataproc
- Pros:
  - Familiar Spark SDK
  - Managed service
  - Lots of integrations
- Cons:
  - Complex streaming model
  - No event time ordering
Solution Trade-offs

Technologies for streaming data

Data Ingestion

Kafka on GCE

- Pros:
  - Simple Producer Interface
  - High throughput
  - Ordering within partitions
- Cons:
  - No REST API
  - Self-managed
  - Scaling complexity

Cloud Pub/Sub

- Pros:
  - Managed global service
  - REST API
  - Pay as you go
- Cons:
  - No ordering of events

Data Processing

Cloud Dataflow

- Pros:
  - Handles unordered data
  - Managed and scalable
  - Integrations with Pub/Sub and Kafka
- Cons:
  - New model for traditional Spark Streaming devs.

Cloud Dataproc

- Pros:
  - Familiar Spark SDK
  - Managed service
  - Lots of integrations
- Cons:
  - Complex streaming model
  - No event time ordering
Batch and Streaming Architecture

- Mobile Users
  - Cloud Firestore
- On-Prem Systems
  - Cloud Pub/Sub
  - Cloud Dataflow
  - Cloud BigQuery
- Compute
- Storage
Recommendation Engine Trade-offs

- **Content Based**: Select most popular products
- **Collaborative Filtering**: Behavior and pattern driven
- **Knowledge Based**: Ask user for preferences
Recommendation Engine Trade-offs

- **Content Based**
  - Select most popular products

- **Collaborative Filtering**
  - Behavior and pattern driven

- **Knowledge Based**
  - Ask user for preferences
Weighted Alternating Least Squares

- Method that uses large matrix factorization to derive similarities between sparse datasets.
- Useful for recommendation systems with very little initial user data like reviews or click data.
- Easy to run in distributed environment and optimize model performance with Tensorflow.
Solution Trade-offs

Technologies for Machine Learning

Model Training
- Tensorflow/MXNet/Caffe on GCE
  - Pros: Control over framework. Good for existing models in other SDKs.
  - Cons: Self-hosted, operational overhead.
- Cloud Machine Learning

Model Serving
- App Engine Flexible
  - Pros: Hosted solution, managed API for Tensorflow models. Integration with Cloud ML
  - Cons: Only supports Tensorflow. API more opinionated
- Cloud Machine Learning

Custom Serving on App Engine
- Pros: Full control over serving API. Security with Endpoints. Easy to deploy.
  - Cons: Different workflow than traditional deployments
Solution Trade-offs

Technologies for Machine Learning

Model Training

Tensorflow/MXNet/Caffe on GCE

- **Pros:**
  - Control over framework.
  - Good for existing models in other SDKs.
- **Cons:**
  - Self-hosted, operational overhead.

Cloud Machine Learning

- **Pros:**
  - Fully managed Tensorflow
  - Automatic config of GPU/CPU.
  - Integrated monitoring
- **Cons:**
  - Only supports Tensorflow

Model Serving

App Engine

- **Pros:**
  - Full control over serving API.
  - Security with Endpoints.
  - Easy to deploy.
- **Cons:**
  - Different workflow than traditional deployments

Custom Serving on App Engine

- **Pros:**
  - Hosted solution, managed API for Tensorflow models.
  - Security with Endpoints.
- **Cons:**
  - Only supports Tensorflow
  - API more opinionated

Cloud Machine Learning

Google Cloud
Recommendation Engine Architecture

Google Cloud Platform

Batch Processing
- Listing Data Cloud Firestore
- Listing Data Cloud Firestore
- Cloud Dataproc

Cloud Storage

Machine Learning
- Tensorflow Training Cloud Machine Learning
- Tensorflow Model Serving App Engine Flexible

Google Cloud

Confidential & Proprietary
Data Processing Architecture

Google Cloud Platform

- Listing/User Data: Cloud Firestore
- Cloud Functions
- Cloud Pub/Sub
- Cloud Dataflow
- Cloud Storage
- Clickstream Data: Cloud BigQuery
- Cloud Dataproc
- Tensorflow Training: Cloud Machine Learning
- Tensorflow Model Serving: App Engine Flexible
Tooling
Some definitions

**Continuous Integration (CI)**
Merge all developer working copies to a shared mainline several times a day

**Continuous Delivery (CD)**
Ensure that your code can be shipped at any time. Requires CI.

**Continuous Deployment (also CD)**
Every change is automatically deployed to production. Requires Continuous Delivery.
CI/CD in practice

Workflows will vary by branch management, staging envs, tests, platform, etc.
CI/CD Tools

- Travis CI
- Circle CI
- Container Builder
- Jenkins
- Spinnaker
- Terraform
- Deployment Manager
- Firebase deploy
CI workflow with Container Builder

- **triggers**
  - GitHub
  - BB
  - CSR
  - Manual
  - CI systems

- **build and test runner**
  - Different stage types, can run any number, in any sequence, and in parallel
  - Custom scripts
  - Bazel builds
  - Non-bazel builds
  - Container jobs
  - CLI commands

- **Result storage + dashboard**
- **3rd party cloud**
- **Build status**
- **Result storage + dashboard**

---

- **current roadmap**
  - Cron
  - Webhook
  - Registry
  - PR
  - Repo dependency

---

- **Google Cloud**
Spinnaker: orchestrate deployment pipelines

Spinnaker Pipeline to promote and release image to production:

1. Start
2. Find image from test
3. Deploy to prod (red/black)
4. Smoke test
5. Scale down old prod
6. Wait 30 mins
7. Tag source and manual approval
8. Wait 2 hrs
9. Destroy old prod
# Solution Trade-offs

<table>
<thead>
<tr>
<th>Solution</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
</table>
| **Pro**  | - Hosted  
- Easy to get started | - Only simple deployment flows are possible  
- Single tool for team to learn |
| **Con**  | | |

## Pro
- Intuitive UI, easy learning curve  
- Integrations with many deployment platforms  
- Streamlined for complex deployments

## Con
- No managed version (yet)  
- Does not support Firebase
## Solution Trade-offs

<table>
<thead>
<tr>
<th><strong>Pro</strong></th>
<th><strong>Con</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>- Hosted</td>
<td>- Only simple deployment flows are possible</td>
</tr>
<tr>
<td>- Easy to get started</td>
<td>- Single tool for team to learn</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Pro</strong></th>
<th><strong>Con</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>- Intuitive UI, easy learning curve</td>
<td>- No managed version (yet)</td>
</tr>
<tr>
<td>- Integrations with many deployment platforms</td>
<td>- Does not support Firebase</td>
</tr>
<tr>
<td>- Streamlined for complex deployments</td>
<td></td>
</tr>
</tbody>
</table>

**Recommendation:** Container Builder
Container Builder build flow

- Dockerfile
  - base image
  - install tools
  - entrypoint

- builder source

- cloudbuild.yaml

- GCS

- source repo

- project source

- build step container
  - build step container
  - build step container
  - ...
Key Questions: Which cloud services? How many resources? What key applications? How much log data do you generate? How long do you retain?

**Pro**
- Native to GCP
- Integrated monitoring/logging
- Fully managed, works at scale
- Trace, Error Reporting, Debug

**Con**
- Switching costs, unfamiliarity

---

**Pro**
- Intuitive UI, easy learning curve
- More integrations
- Cloud-based, managed service

**Con**
- Monitoring only, need separate solution for logging
- Higher licensing cost

---

**Pro**
- Familiar, industry standard
- Advanced log search functionality
- Hybrid: 1 UI for on-prem/cloud
- [Splunk Add-On for GCE](#)

**Con**
- Logging only
- Self-hosted
- Higher TCO cost
# Logging/Monitoring

**Key Questions:** Which cloud services? How many resources? What key applications? How much log data do you generate? How long do you retain?

<table>
<thead>
<tr>
<th><strong>Pro</strong></th>
<th><strong>Con</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Native to GCP</td>
<td>Switching costs, unfamiliarity</td>
</tr>
<tr>
<td>Integrated monitoring/logging</td>
<td>- Monitoring only, need separate solution for logging</td>
</tr>
<tr>
<td>Fully managed, works at scale</td>
<td>- Higher TCO cost</td>
</tr>
<tr>
<td>Trace, Error Reporting, Debug</td>
<td>- Self-hosted</td>
</tr>
</tbody>
</table>

**Pro**
- Intuitive UI, easy learning curve
- More integrations
- Cloud-based, managed service

**Con**
- Monitoring only
- Advanced log search functionality
- Hybrid: 1 UI for on-prem/cloud
- Splunk Add-On for GCE

**Recommendation:** Stackdriver
## IAM/Admin

### Key questions: What is the source of truth for your user credentials today? Require separate IT and Dev access?

<table>
<thead>
<tr>
<th>Using On-Prem LDAP</th>
<th>Google Authentication</th>
<th>Using SSO (e.g. Okta)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pro</strong></td>
<td><strong>Pro</strong></td>
<td><strong>Pro</strong></td>
</tr>
<tr>
<td>- Can provision accounts (one-way) with GCDS - no programming</td>
<td>- No LDAP or sync to manage</td>
<td>- Integrates with other tools</td>
</tr>
<tr>
<td>- Can sync passwords (PasswordSync)</td>
<td>- Google-stored credentials: Google manages authentication, stores the passwords</td>
<td>- No additional authentication to set up</td>
</tr>
<tr>
<td><strong>Con</strong></td>
<td><strong>Con</strong></td>
<td><strong>Con</strong></td>
</tr>
<tr>
<td>- Have to manage own LDAP</td>
<td>- Google two-factor authentication</td>
<td>- Single point of failure</td>
</tr>
<tr>
<td>- Requires LDAP admin skills + access</td>
<td>Google-only, does not integrate with other tools for SSO</td>
<td>- Additional system to manage</td>
</tr>
</tbody>
</table>
IAM/Admin

Pro
- Can provision accounts (one-way) with GCDS - no programming
- Can sync passwords (PasswordSync)

Con
- Have to manage own LDAP
- Requires LDAP admin skills + access

Using on-prem LDAP

Moving to Google Accounts

Pro
- No LDAP or sync to manage
- Google-stored credentials: Google manages authentication, stores the passwords
- Google two-factor authentication

Con
Google-only, does not integrate with other tools for SSO

Using SSO (e.g. Okta)

Pro
- Integrates with other tools
- No additional authentication to set up
- Can enforce MFA, custom security controls

Con
- Single point of failure
- Additional system to manage
- No advanced Google security features

Key questions: What is the source of truth for your user credentials today? Require separate IT and Dev access?

Recommendation: Cloud Identity, Using on-prem LDAP with Google Auth
**Deployment**

**Key Questions**: Are you on multiple platforms today? Do you have an Ops team to run a self-hosted service? Which deployment tools do you use today?

**Deployment Manager**

**Pro**
- Integrated with GCP - UI, IAM, labels
- Managed service, No Ops
- New GCP functionality arrives immediately
- Declarative, immutable

**Con**
- GCP only
- Need a separate platform for multiple clouds

**Pro**
- Multi-cloud
- Familiarity
- Declarative, immutable

**Con**
- Non-native to GCP
- Lag time to get new GCP functionality
- Self-hosted -> manage + pay for a server
- Not supported by GCP, no SLA

**Manual scripting**

**Pro**
- Easier to get started
- No new system to learn or install

**Con**
- Imperative, mutable
- Error-prone, toil intensive
- Cannot roll back easily
- Hard to automate
Deployment

**Key Questions:** Are you on multiple platforms today? Do you have an Ops team to run a self-hosted service? Which deployment tools do you use today?

**Deployment Manager**

**Pro**
- Integrated with GCP - UI, IAM, labels
- Managed service, No Ops
- New GCP functionality arrives immediately
- Declarative, immutable

**Con**
- GCP only
- Need a separate platform for multiple clouds

**Terraform**

**Pro**
- Multi-cloud
- Familiarity
- Declarative, immutable

**Con**
- Non-native to GCP
- Lag time to get new GCP functionality
- Self-hosted -> manage + pay for a server
- Not supported by GCP, no SLA

**Manual scripting**

**Pro**
- Easier to get started
- No new system to learn or install

**Con**
- Imperative, mutable
- Error-prone, toil intensive
- Cannot roll back easily
- Hard to automate

Recommendation: Deployment Manager
Strategy - Forseti Open Source Toolkit

Google’s own cloud security tools; built and shared with everyone

**Inventory**
- Build and store an inventory
  - Projects
  - Roles
  - Members
  - ....

**Policy Scanning**
- Find policy violations
  - Organization level
  - Project level
  - ....

**Enforcement**
- Apply changes
  - Inbound GCE firewall
  - ....
Security

**Key Questions:** Do you have the skill set and resources to write your own security tooling flow? Do you need custom tooling that isn't provided by Forseti?

**Forseti**

**Pro**
- Open Source - based on Google’s own tools, shared with everyone
- Integrated suite
- Google (and community) investing heavily - new features on the way

**Con**
Less customization

**Custom Tooling**

**Pro**
- More customizable
- Can still use Google managed services (Audit logs -> Pub/Sub -> Cloud Functions)

**Con**
Have to code, configure and deploy yourself

**No Tooling**

**Pro**
No work to be done

**Con**
Pretty insecure...
**Security**

**Key Questions:** Do you have the skill set and resources to write your own security tooling flow? Do you need custom tooling that isn't provided by Forseti?

<table>
<thead>
<tr>
<th>Forseti</th>
<th>Custom Tooling</th>
<th>No Tooling</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pro</strong></td>
<td><strong>Pro</strong></td>
<td><strong>Pro</strong></td>
</tr>
<tr>
<td>- Open Source - based on Google’s own tools, shared with everyone</td>
<td>- More customizable</td>
<td>- No work to be done</td>
</tr>
<tr>
<td>- Integrated suite</td>
<td>- Can still use Google managed services (Audit logs -&gt; Pub/Sub -&gt; Cloud Functions)</td>
<td></td>
</tr>
<tr>
<td>- Google (and community) investing heavily - new features on the way</td>
<td>- Have to code, configure and deploy yourself</td>
<td></td>
</tr>
<tr>
<td><strong>Con</strong></td>
<td><strong>Con</strong></td>
<td><strong>Con</strong></td>
</tr>
<tr>
<td>Less customization</td>
<td>Have to code, configure and deploy yourself</td>
<td>Pretty insecure...</td>
</tr>
</tbody>
</table>

**Recommendation:** Forseti
Google is an AI company

Google internal projects containing Brain Model

Unique project directories

Time

Used across products:
Very few users today can create a custom ML model. To democratize AI, we need to make AI accessible to millions more.
The popular imagination of what ML is

Lots of data  →  Complex mathematics in multidimensional spaces  →  Magical results
The Reality of ML: Complex & Time Intensive

Large computational resource .  Machine learning expertise .  Manual data labeling
Cloud AutoML - automatically create Machine Learning Models

DATA PREPROCESSING → ML MODEL DESIGN → TUNE ML MODEL PARAMETERS → EVALUATE → DEPLOY → UPDATE

Simple transfer learning with efficient model training for quick demo within minutes
High accuracy model based upon learning to learn within a day
Google’s Machine Learning Ecosystem

Perception APIs
Google-trained models for your app

Auto ML
Easy to use, composable, learn to learn technology

Training & Prediction
End to End Platform for your data & models

App Developers
Analysts
Data Engineers / Data Scientists

Using Machine Learning to Explore Neural Network Architecture - Quoc Le & Barret Zoph, Research Scientists, Google Brain team
Model is now trained and ready to make prediction. This model can scale as needed to adapt to customer demands.
Serverless = No Ops, Right??

✓ Scalable Ops
Shrinking The Scaling Point

- Your Code
- Your Code
- Your Code
- Your Code
- Fns
- PaaS
- Containers
- Virtual Machines
- Dedicated Physical Servers