NobleProg

Developing Solutions with GCP

Presented by Bin Wu

The World's Local Training Provider

GCP benefits

- Developer friendly
  - Web console built-in command-line
  - API services playground
  - Step-by-step setup documents
- End user to development ecosystem (semi-mature)
  - Google Apps (G'drive, docs, mail etc.)
  - Cloud Platform
  - Data Analytics Suite (GA, DataStudio, GTM etc.)
- AI
  - TensorFlow
  - Encapsulated ML APIs

NobleProg

Dev tools and services for GCP

- Cloud SDK
  - Command-line interface for Google Cloud Platform products and services
  - Manage all of your Google Cloud Platform projects from the command line including compute, networking, storage and development products.
- Cloud Shell
  - Manage your infrastructure and applications from the command-line in any browser
  - Your own Linux VM accessible from your browser allows you to manage your GCP resources with all of the necessary tools pre-installed and up-to-date.
- Cloud Source Repositories
  - Private Git repositories hosted on Google Cloud Platform
  - Enable collaborative development with hosted Git repositories and improve developer productivity with automatic integration with Stackdriver Debugger and other Google Cloud Platform diagnostic tools.

NobleProg

Dev tools and services for GCP - cont'd

- Cloud Tools for Android Studio
  - Build backend services on Google Cloud Platform for your Android apps
  - Google Cloud Platform plugin for Android Studio lets you unify the development experience by letting you code, build, test and validate your backends in the same environment you use to develop your mobile apps.
- Cloud Tools for IntelliJ
  - Deploy and debug production cloud applications right inside of IntelliJ
  - Google Cloud Platform plugin for IntelliJ lets you easily deploy and debug your Java applications directly from your IDE.
- Cloud Tools for Eclipse
  - Deploy and debug App Engine Standard applications right inside of Eclipse
  - Google Cloud Platform plugin for Eclipse lets you easily deploy and debug your Java App Engine Standard applications directly from your IDE.

NobleProg

Dev tools and services for GCP - cont'd

- Cloud Tools for PowerShell
  - Full Google Cloud Platform control from Windows PowerShell
  - Cloud Tools for PowerShell lets you script, automate, and manage your Google Cloud Platform projects.

- Cloud Tools for Visual Studio
  - Deploy Visual Studio applications to Google Cloud Platform
  - Google Cloud Platform plugin for Visual Studio lets you easily deploy your .NET applications directly and manage your projects from Visual Studio.

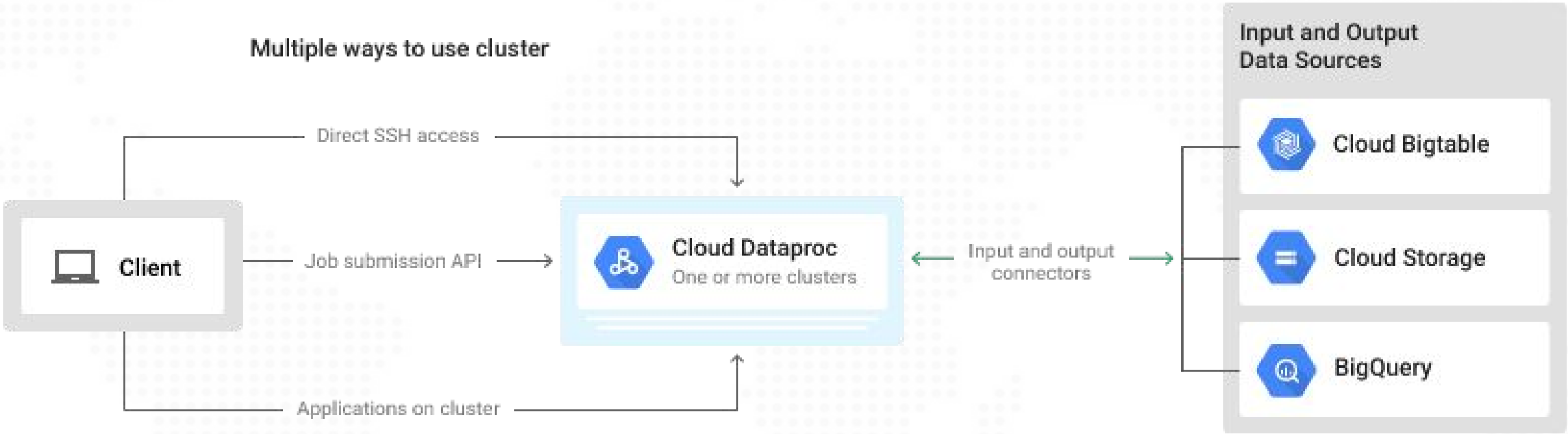
Dev tools and services for GCP - cont'd

- Cloud Deployment Manager
  - Create and manage cloud resources with simple templates
  - Google Cloud Deployment Manager allows you to specify all the resources needed for your application in a declarative format using yaml.
- Maven App Engine Plugin
  - Build and deploy your App Engine apps using Maven
  - The Google Maven Plugin and archetypes lets you easily build and deploy your App Engine applications using Maven.
- Gradle App Engine Plugin
  - Build and deploy your App Engine apps using Gradle
  - The Google Gradle Plugin lets you easily build and deploy your App Engine applications using Gradle.

Dev tools and services for GCP - cont'd

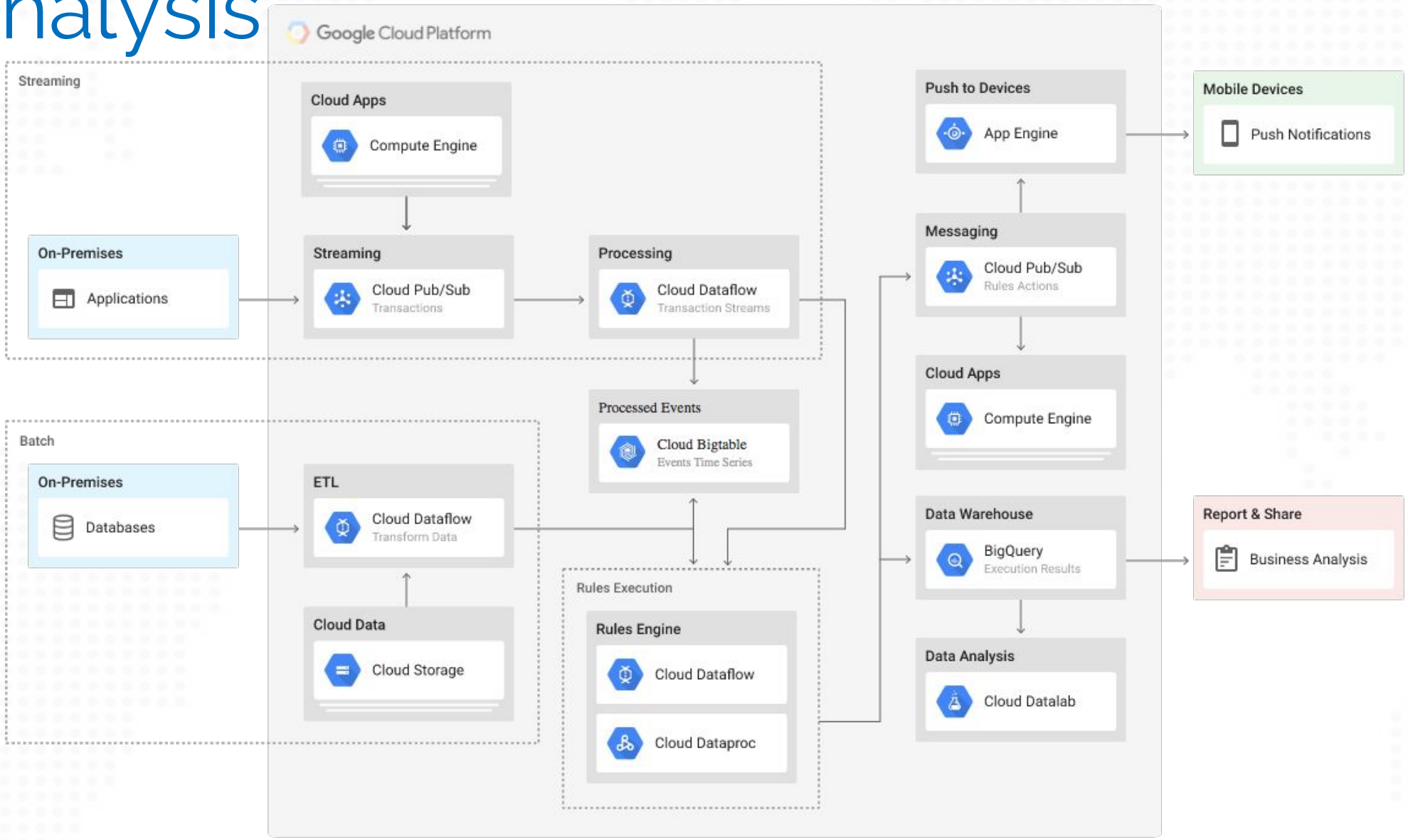
- Google Analytics Query Explorer
  - This tool lets you play with the Core Reporting API by building queries to get data from your Google Analytics views (profiles). You can use these queries in any of the client libraries to build your own tools.
- Google OAuth2.0 Playground
  - Test and develop authentication and authorization related functions
- Google APIs Exlorer
  - The Explorer lets you select a supported service and view a list of supported methods. You can then select a method, fill in the parameters, execute the command and get the results. It seems like a great way to try out an API without writing code.

SA - Machine Learning



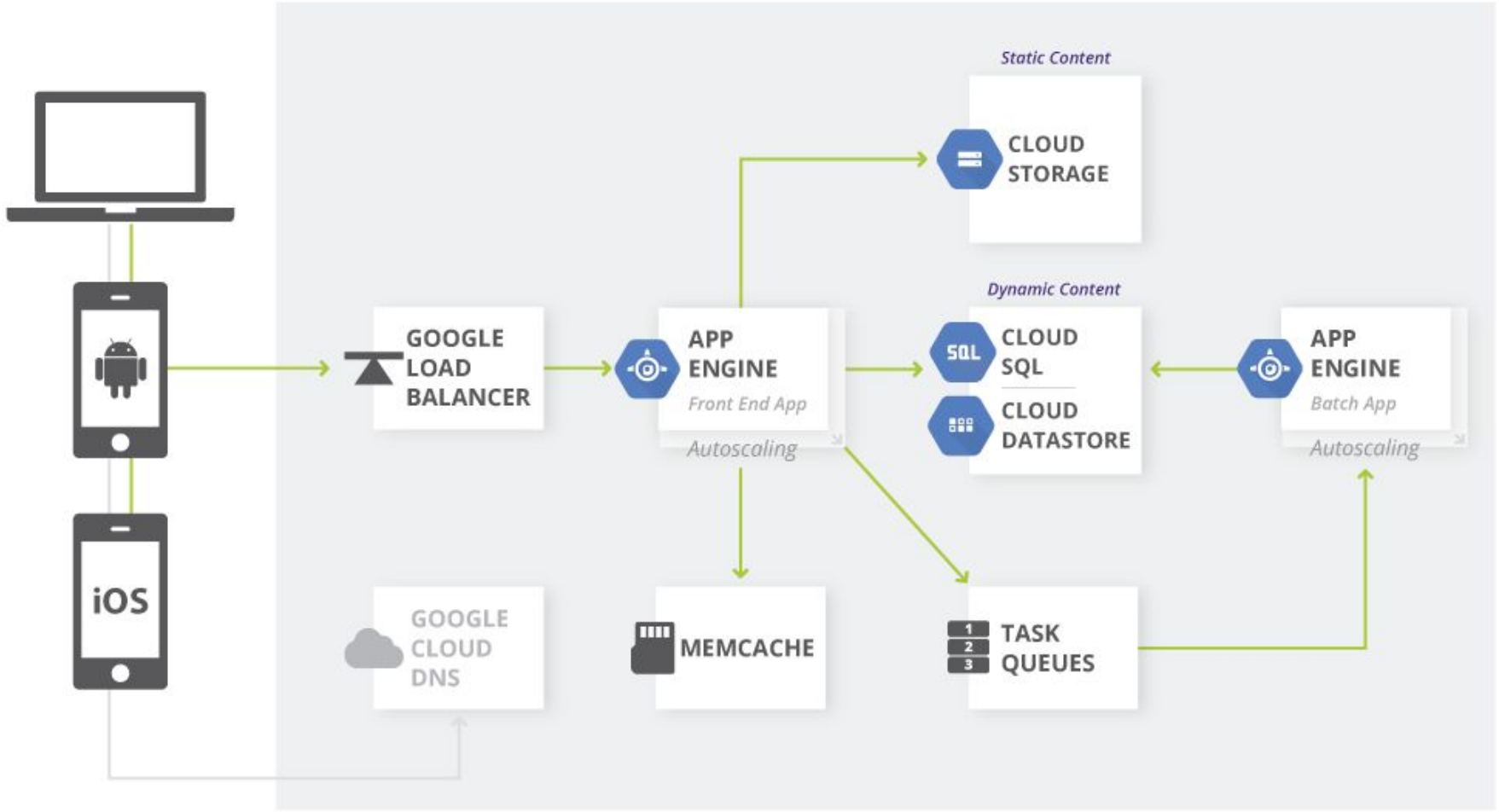
NobleProg

SA - Analysis



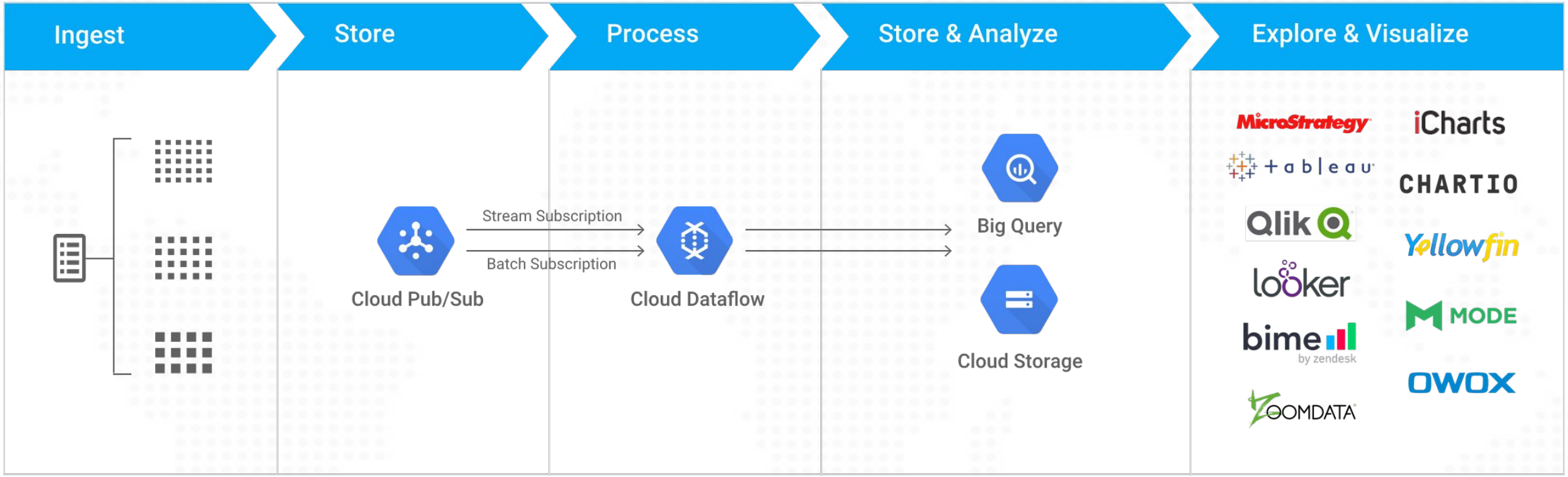
NobleProg

SA - webapp



NobleProg

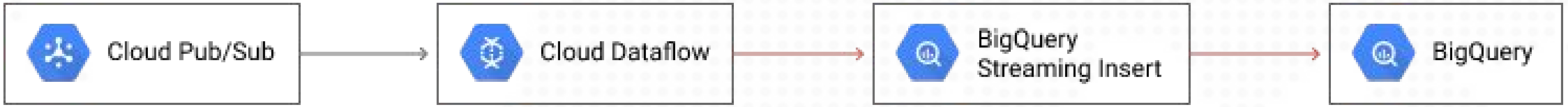
SA - Clickstream Analysis



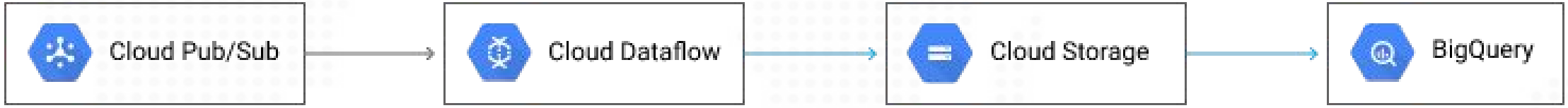
NobleProg

SA - Large-Scale Ingestion of Analytics Events and Logs

Hot data



Cold data



NobleProg

Lab - Source Repositories

- Create repo
  - *gcloud init*
  - *gcloud source repos create nobleprog*
- Checkout
  - *gcloud source repos clone nobleprog*
- Use git commands to push master
- Examine files in code repor

NobleProg

Cloud Endpoints

- An NGINX-based proxy and distributed architecture give unparalleled performance and scalability
- Access control with JWT and Google API keys
- Fast
  - Extensible Service Proxy delivers security and insight in less than 1ms per call
  - Deploy your API automatically with Google App Engine
- Open source Cloud Endpoints Frameworks in Java or Python

NobleProg

Cloud Endpoints - Dev process

1. Write your API code first, wrapping the classes and any methods to be exposed, and creating message classes
2. Create an API server
3. Generate the OpenAPI configuration file and deploy it
4. Deploy the API.
5. Generate Client Libraries.
6. Write your client app, using the client libraries when making calls to the API.

NobleProg

Cloud Endpoints - Create API

Standard imports

```
import endpoints
from protorpc import message_types
from protorpc import messages
from protorpc import remote
```

Derive from framework

```
@endpoints.api(name='echo', version='v1')
class EchoApi(remote.Service):
```

Handler

```
@endpoints.method(
    # This method takes a ResourceContainer defined above.
    message_types.VoidMessage,
    # This method returns an Echo message.
    EchoResponse,
    path='echo/getApiKey',
    http_method='GET',
    name='echo_api_key')
def echo_api_key(self, request):
    return EchoResponse(content=request.get_unrecognized_field_info('key'))
```


NobleProg

Cloud Endpoints - API Server

API Server (main.py)

```
api = endpoints.api_server([EchoApi])
```

Map the API Server (app.yaml)

```
handlers:
# The endpoints handler must be mapped to /_ah/api.
- url: /_ah/api/.*
  script: main.api
```


NobleProg

Cloud Endpoints - Generate OpenAPI conf

Generate OpenAPI conf

```
appengine_config.py  app.yaml  lib  main.py  main.pyc  main_test.py  README.md  requirements.txt
[redacted]@refined-network-179706:~/workspace/python-docs-samples/appengine/standard/endpoints-fr[redacted] python lib/endpoints/endpointscfg.py get_openap
i_spec main.EchoApi --hostname refined-network-179706.appspot.com
No handlers could be found for logger "oauth2client.contri[redacted].multistore_file"
WARNING:root:Method echo.echo_path_parameter specifi[redacted]path parameters but you are not using a ResourceContainer. This will fail in future releases; please switch
 to using ResourceContainer as soon as possibl[redacted]
OpenAPI spec written to ./echov1openapi.json
```

echov1openapi.json

```
"/echo/v1/echo/getApiKey": {
  "get": {
    "operationId": "EchoApi_echoApiKey",
    "parameters": [],
    "responses": {
      "200": {
        "description": "A successful response",
        "schema": {
          "$ref": "#/definitions/MainEchoResponse"
        }
```


NobleProg

Cloud Endpoints - deploy API

- Deploy the endpoints config
  - *gcloud service-management deploy echov1openapi.json*

```
Service Configuration [2017-10-15r0] uploaded for service [refined-network-179706.appspot.com]

To manage your API, go to: https://console.cloud.google.com/endpoints/api/refined-network-179706.appspot.com/overview?project=refined-network-179706
```

- Double check service config
  - *gcloud service-management configs list --service=refined-network-179706.appspot.com*

```
CONFIG_ID      SERVICE_NAME
2017-10-15r0   refined-network-179706.appspot.com
```

- Deploy backend
  - *gcloud app deploy*

NobleProg

Lab - Cloud Endpoints

- Endpoints on Compute Engine
  - Follow the instructions to setup endpoints backend on Compute Engine
- Use JavaScript client to consume endpoints APIs

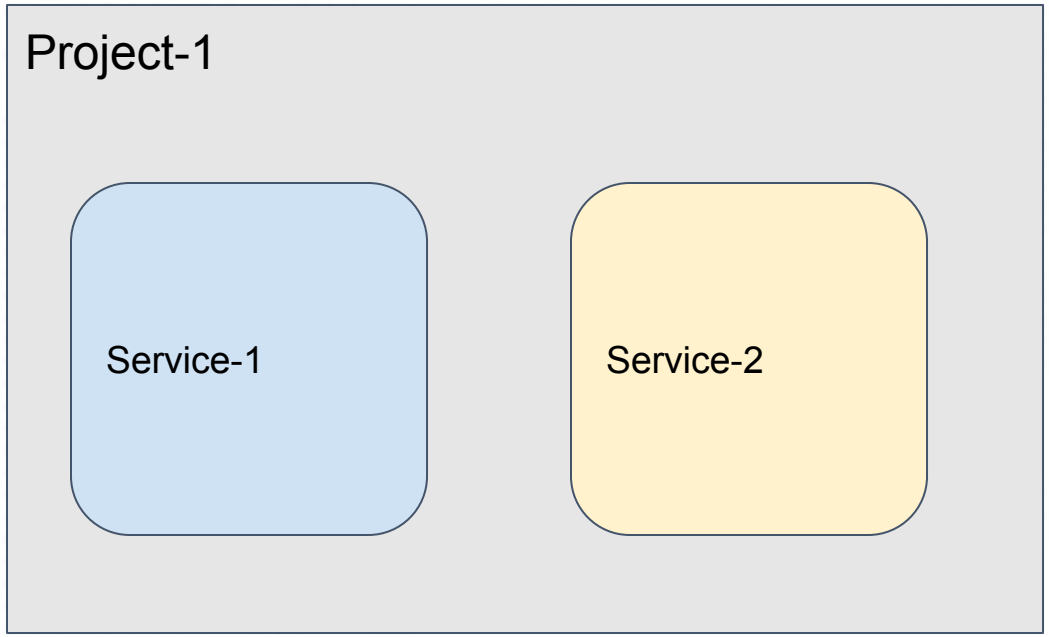
NobleProg

Google App Engine - Microservices

- Define strong contracts between the various microservices.
- Allow for independent deployment cycles, including rollback.
- Facilitate concurrent, A/B release testing on subsystems.
- Minimize test automation and quality-assurance overhead.
- Improve clarity of logging and monitoring.
- Provide fine-grained cost accounting.
- Increase overall application scalability and reliability.
- GraphQL

App Engine Service as Microservice

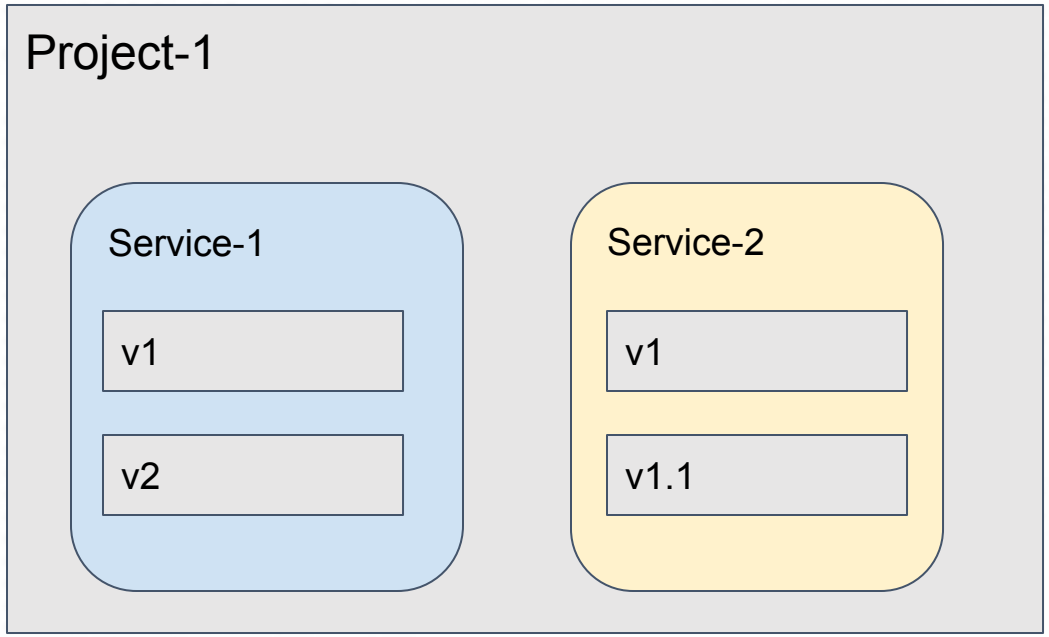
- Independent
  - Code
  - Autoscaling
  - Load balancing
- Different languages
  - Python, java, node.js



NobleProg

Versions in App Engine Services

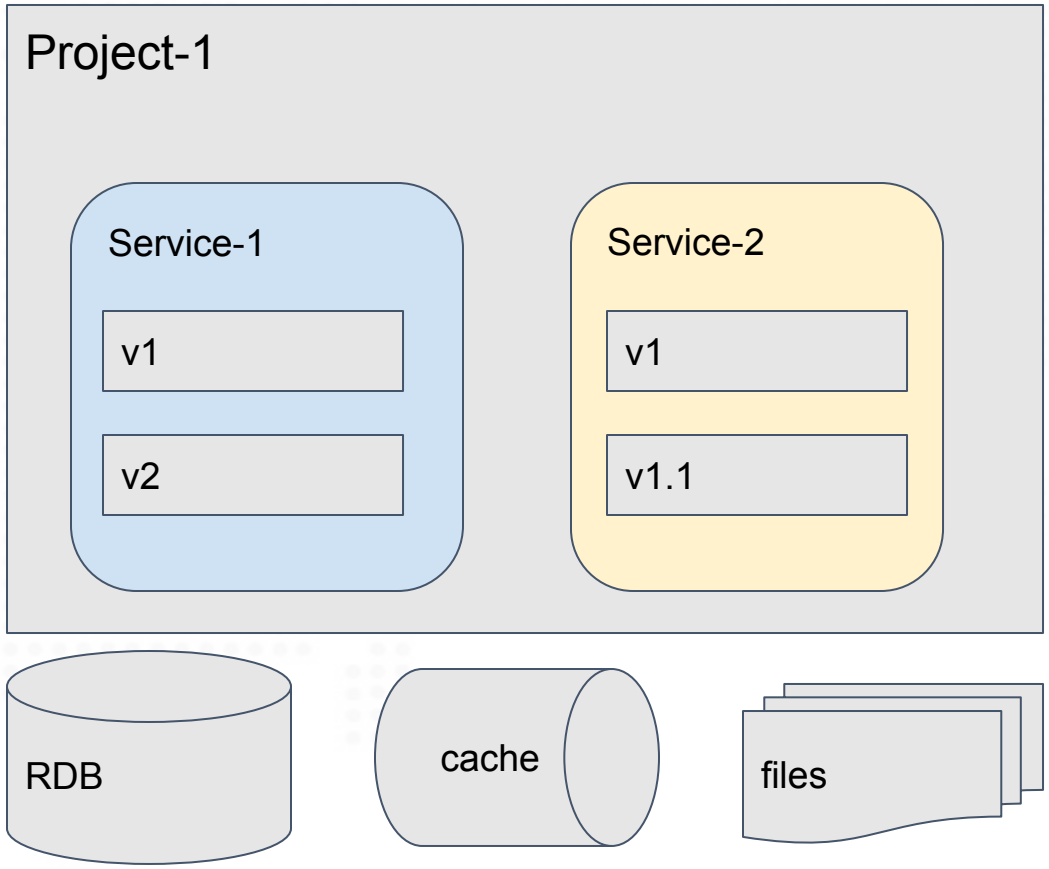
- Multiple versions
  - One default
  - A/B testing
  - Roll-forward
  - rollback



NobleProg

Isolation

- Services
  - Isolated services
  - Shared data storages
- Projects
  - Multiple app engine projects
- Hybrid



NobleProg

Isolation - cont'd

	Multiple services	Multiple projects
Code isolation	Deployed code is completely independent between services and versions.	Deployed code is completely independent between projects, and between services and versions of each project.
Data isolation	Cloud Datastore and Memcache are shared between services and versions, however *namespaces* can be used as a developer pattern to isolate the data. For Task Queue isolation, a developer convention of queue names can be employed, such as **`user-service-queue-1`**.	Cloud Datastore, Memcache, and Task Queues are completely independent between projects.
Log isolation	Each service (and version) has independent logs, though they can be viewed together.	Each project (and service and version of each project) has independent logs, though all the logs for a given project can be viewed together. Logs across multiple projects cannot be viewed together.

NobleProg

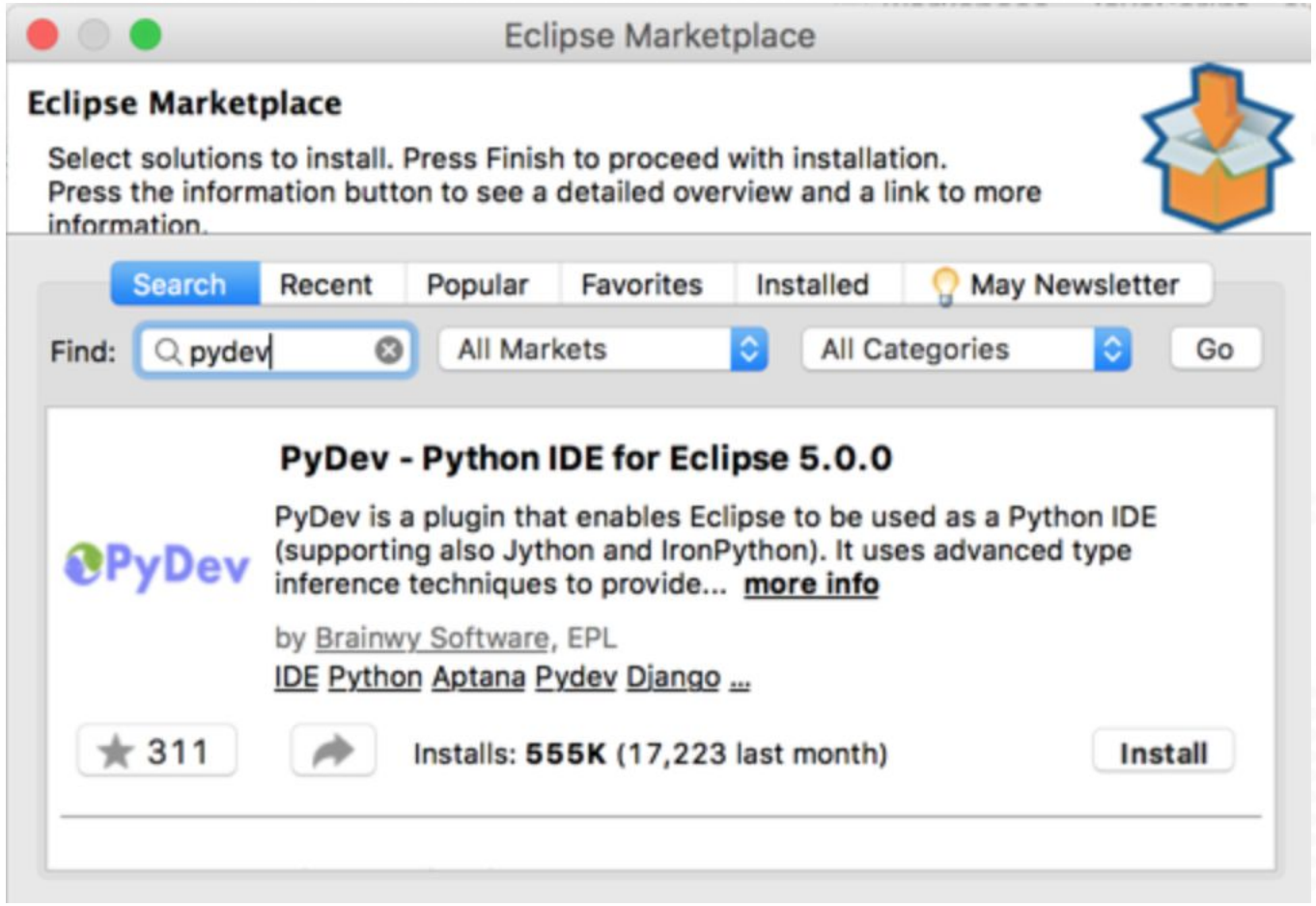
Isolation - cont'd

	Multiple services	Multiple projects
Performance overhead	Services of the same project are deployed in the same datacenter, so the latency in calling one service from another by using HTTP is very low.	Projects might be deployed in different datacenters, so HTTP latencies could be higher, though still quite low because Google's network is world-class.
Cost accounting	Costs for instance-hours (the CPU and memory for running your code) are not separated for services; all the instance-hours for an entire project are lumped together.	Costs for different projects are split, making it very easy to see the cost of different microservices.
Operator permissions	An operator has the ability to deploy code, roll forward and roll back versions, and view the logs for all services of a project. There is no way to limit access to specific services.	Operator access can be controlled separately on separate projects.
Request tracing	Using Google Cloud Trace, you can view a request and the resulting microservice requests for services in the same project as a single composed trace. This feature can help make performance tuning easier.	Cloud Trace calls do not span multiple projects, so end-to-end performance tuning can be more difficult.

NobleProg

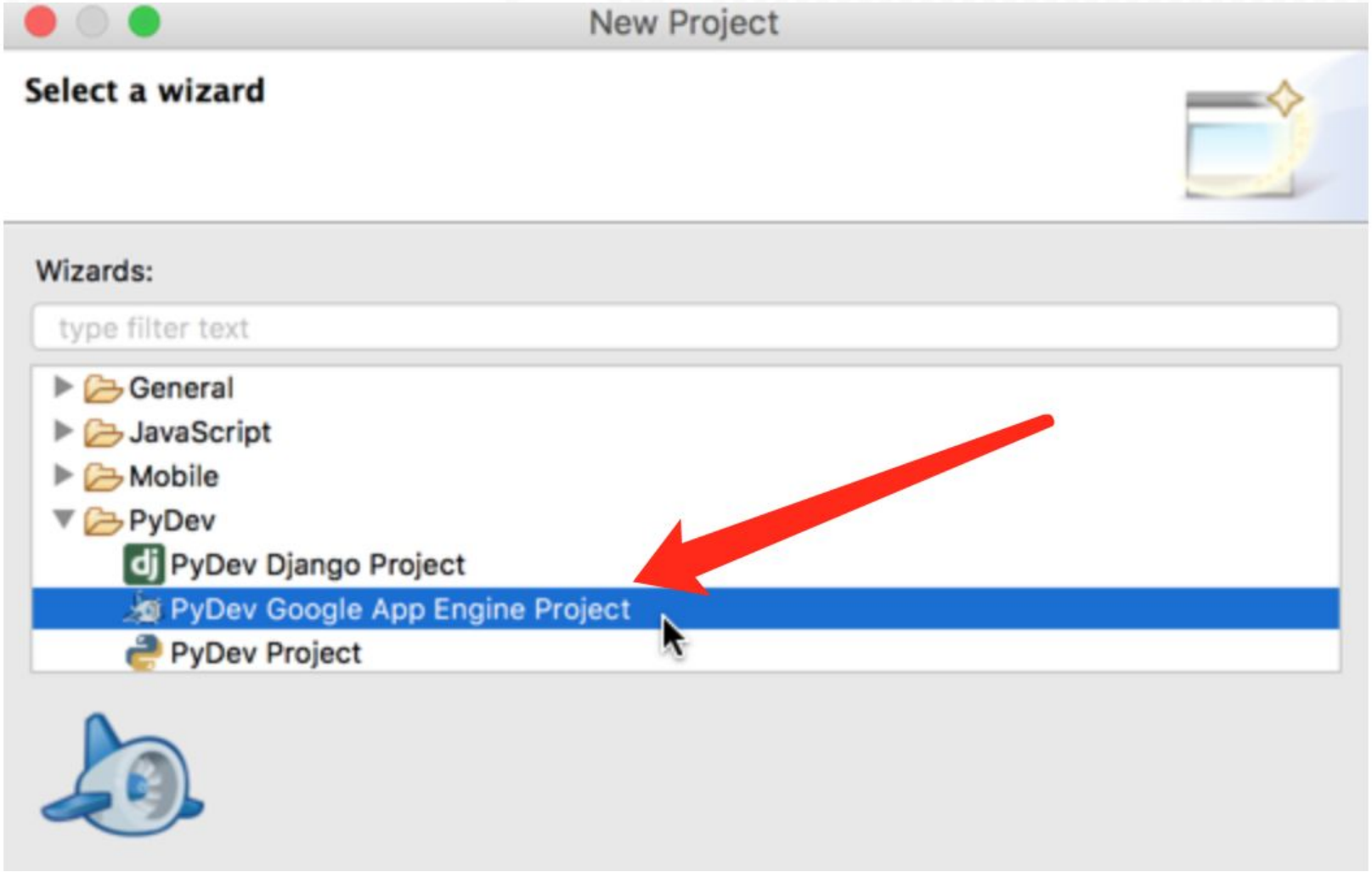
App Engine Dev Environment - python

- Python - Anaconda
- SDK
  - Local dev environment
- IDE - eclipse
  - PyDev
    - Help -> Eclipse Marketplace



NobleProg

Project conf in Eclipse



NobleProg

Project conf in Eclipse - cont'd

- Choose python interpreter (2.7)
- Set SDK path

NobleProg

App Engine local dev environment

- App Engine SDK for python enforces some restrictions
  - Restricted system functions
  - Python module imports
  - No requests timeout and quotas
- Simulates
  - Datastore
  - Memcache
  - Task queues

NobleProg

Lab - App Engine Services

- Deploying Spring boot application in App Engine

Or

- App Engine in Python

NobleProg

Authentication Overview

- Authentication
  - The process of determining one's identity
  - Username & password
- Authorization
  - The process of determining what permissions an authenticated client has for a set of resources
  - OAuth2.0
    - Client id and credentials
    - A protocol without details

NobleProg

GCP authentication

- Service Account
  - Google account that represents an application
  - All GCP APIs supported
- User Account
  - OAuth2.0
- API Key
  - Encrypted string that identifies a Google project for quota and billing purposes
  - Calling endpoints APIs (not recommended, user service account)

NobleProg

Client lib credentials

ADC - Application Default Credentials

- First, ADC checks to see if the environment variable GOOGLE_APPLICATION_CREDENTIALS is set. If the variable is set, ADC uses the service account file that the variable points to.
- If the environment variable isn't set, ADC uses the default service account that Compute Engine, Container Engine, App Engine, and Cloud Functions provide, for applications that run on those services.
- If ADC can't use either of the above credentials, an error occurs.

NobleProg

Client lib credentials - implicit

```
def implicit():
    from google.cloud import storage

    # If you don't specify credentials when constructing the client, the
    # client library will look for credentials in the environment.
    storage_client = storage.Client()

    # Make an authenticated API request
    buckets = list(storage_client.list_buckets())
    print(buckets)
```


NobleProg

Client lib credentials - explicit

```
def explicit():
    from google.cloud import storage

    # Explicitly use service account credentials by specifying the private key
    # file. All clients in google-cloud-python have this helper, see
    # https://google-cloud-python.readthedocs.io/en/latest/core/modules.html
    #   #google.cloud.client.Client.from_service_account_json
    storage_client = storage.Client.from_service_account_json(
        'service_account.json')

    # Make an authenticated API request
    buckets = list(storage_client.list_buckets())
    print(buckets)
```


NobleProg

Authorization - end user authentication

- You need to access resources on behalf of an end user of your application. For example, your application needs to access Google BigQuery datasets that belong to users of your application.
- You need to authenticate as yourself instead of your application. For example, because the Cloud Resource Manager API can create and manage projects owned by a specific user, you would need to authenticate as a user to create projects on their behalf.

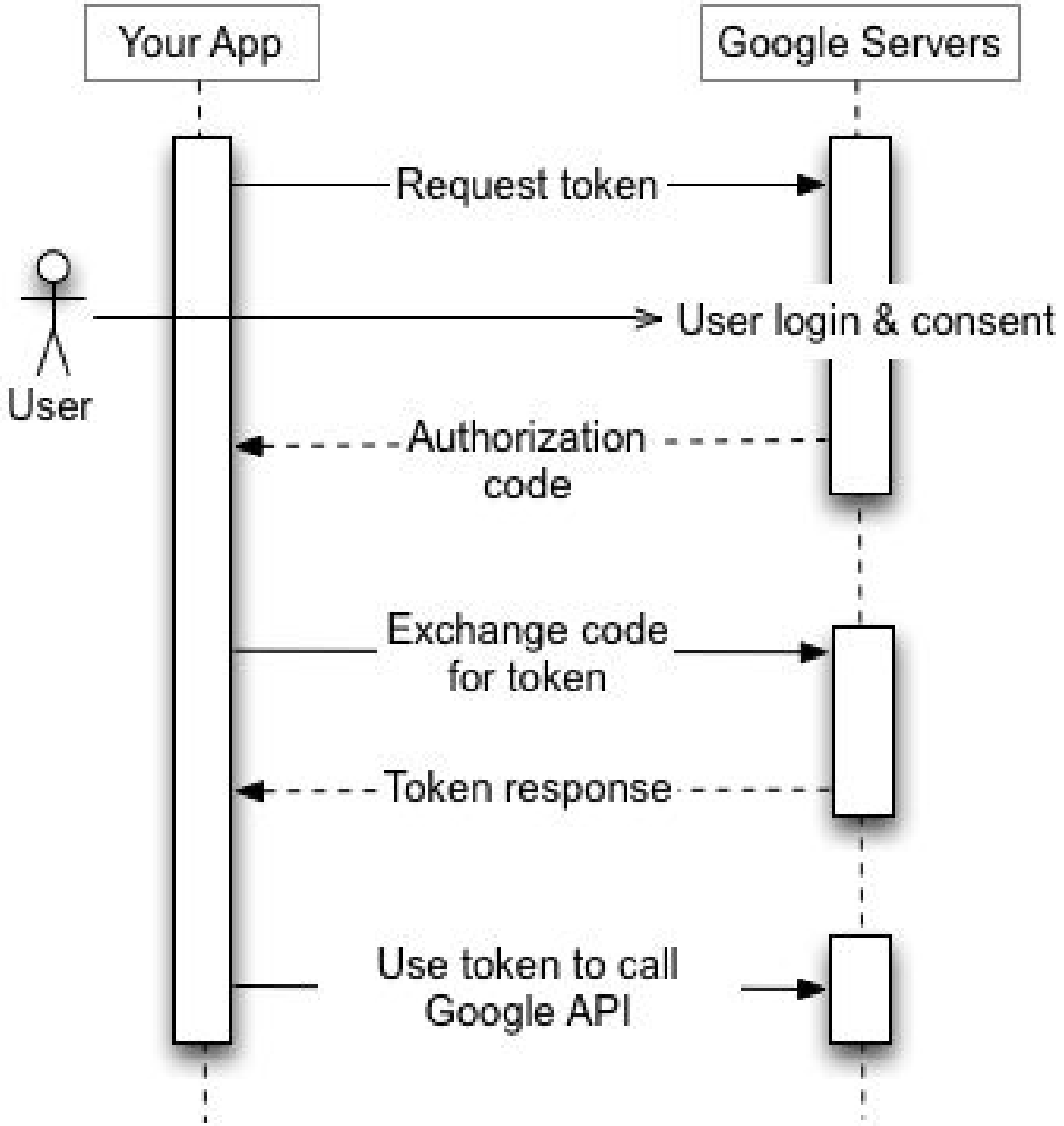
NobleProg

Authorization protocol - OAuth2.0

1. Obtain OAuth 2.0 credentials (Client ID and Client secret).
2. Obtain an access token from the Google Authorization Server.
3. Send the access token to an API.
4. Refresh the access token, if necessary.

NobleProg

OAuth2.0

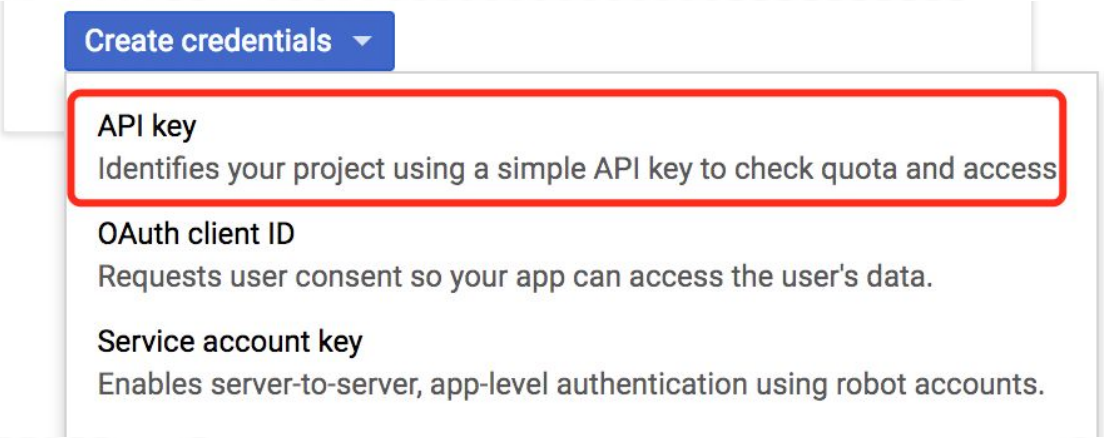


NobleProg

API Key

Application passes this key into all API requests as a key=API_key parameter

- Go to the Cloud Platform Console.
- From the projects list, select a project or create a new one.
- If the APIs & services page isn't already open, open the left side menu and select APIs & services.
- On the left, choose Credentials.
- Click Create credentials and then select API key.



NobleProg

Lab - User Authentication

Using OAuth 2.0 with the Google API Client Library for Java

NobleProg

NobleProg