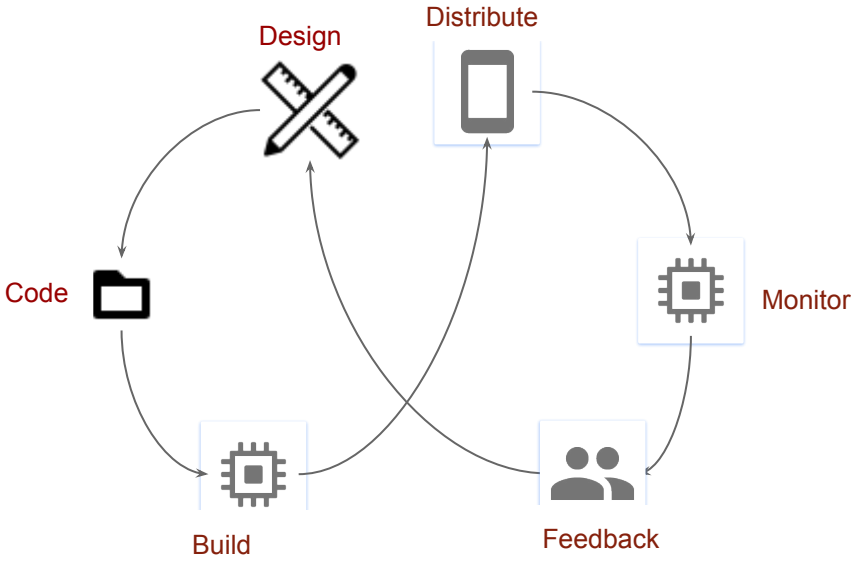
O'reilly Velocity
June, 2018

Serverless Mobile Delivery Pipeline

Google Cloud

This is a talk about Mobile DevOps



Google Cloud

Mobile DevOps

1. **DevOps**: Automates the processes between software development and IT teams, in order that they can build, test, and release software faster and more reliably

2. **Mobile**: means we have to do some extra stuff (beyond what we would do with a web application or api/ microservice) to get the application deployed

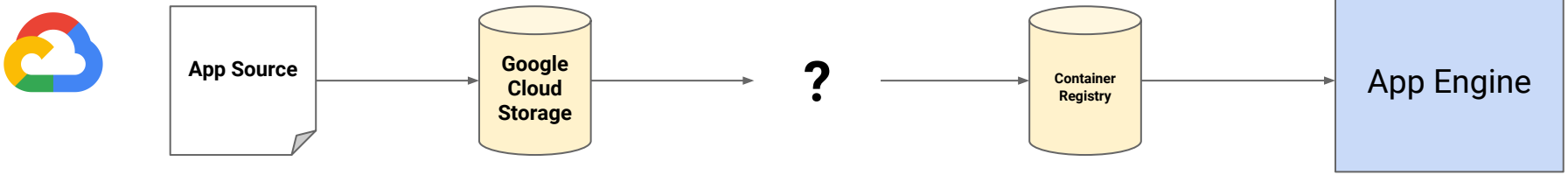
Getting Mobile App Distributed

1. Code the application
2. Push the code to a source code repository
3. Compile the Code
4. Test the Code
5. Test the Application
6. Build the Application Artifact (APK)
7. Distribute the Application Artifact (APK) to your users

Container Builder

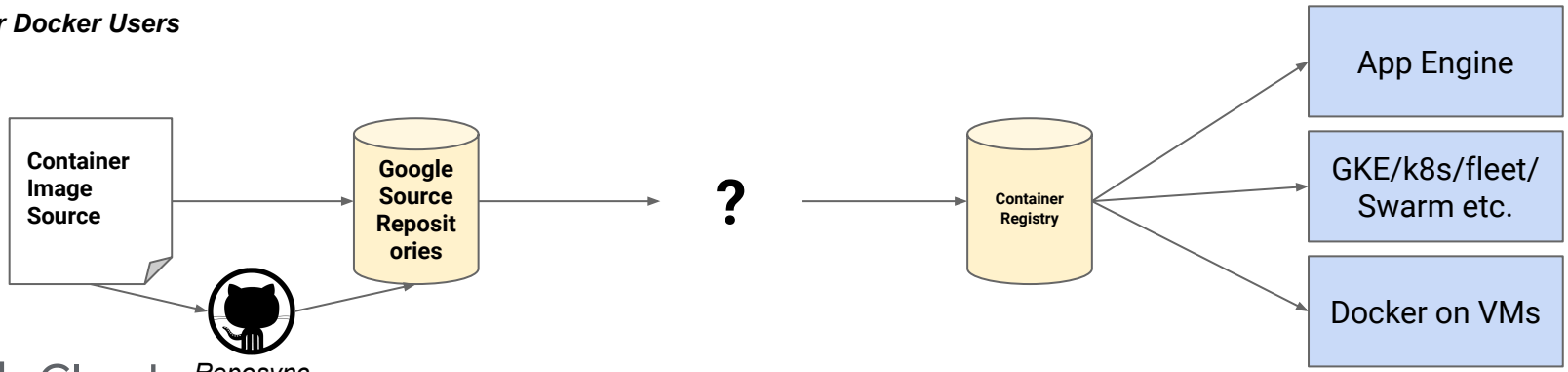
An API providing fast, cost effective and reliable Docker image builds.

Container Builder



App Engine Managed VMs

Other Docker Users



Google Cloud

Fast Cost Effective Docker Builds?

Google Cloud

Container Builder

An API providing fast, cost effective and reliable ~~Docker image builds.~~ Builds

Google Cloud

Docker and Build Dependencies

Container Builder: Not just for Docker Builds

- It can build anything
- It encapsulates environmental dependencies using docker images
- These docker images are called builders
- Many builders are available from the community as well as from Google

Container Builder: Not just for Docker Builds

- Builders can be used to
  - Deploy manifests to Kubernetes/Kubernetes engine
  - Issue gcloud commands,
  - Deploy helm charts
  - Run packer and **much more!**
  - Including …
  - **Building Android Apps!**

Google Cloud

Google Cloud Source Repositories

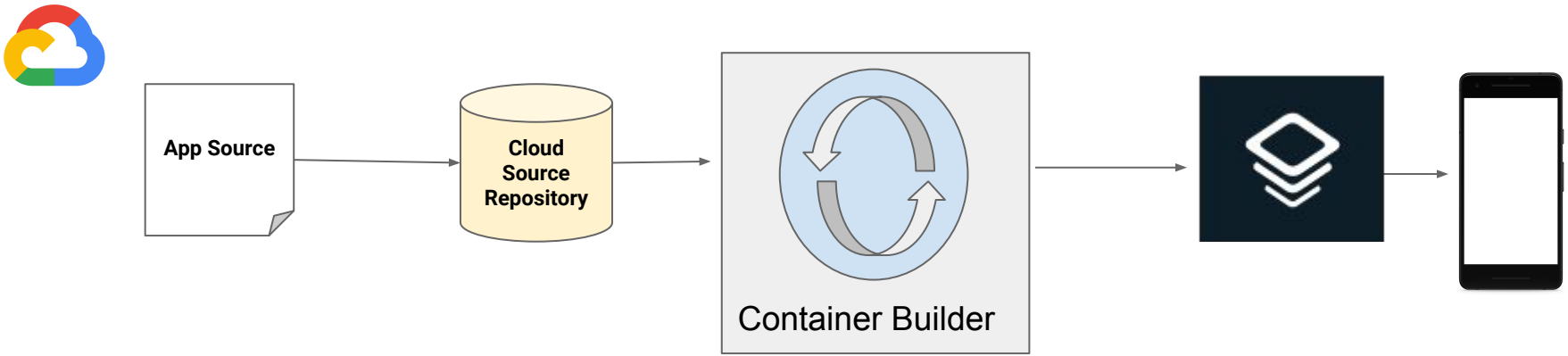
- Git repos in GCP
- Support Build Triggers for Container Builder

Fabric

- Acquired by Firebase in 2017
- Delivery part of the delivery pipeline
- Crashlytics Beta for beta distribution
- Crashlytics monitoring
- Other stuff...

Google Cloud

Container Builder



O’reilly Velocity
March, 2018

Let’s Walk Through Setting Up Our App Delivery Pipeline on GCP!

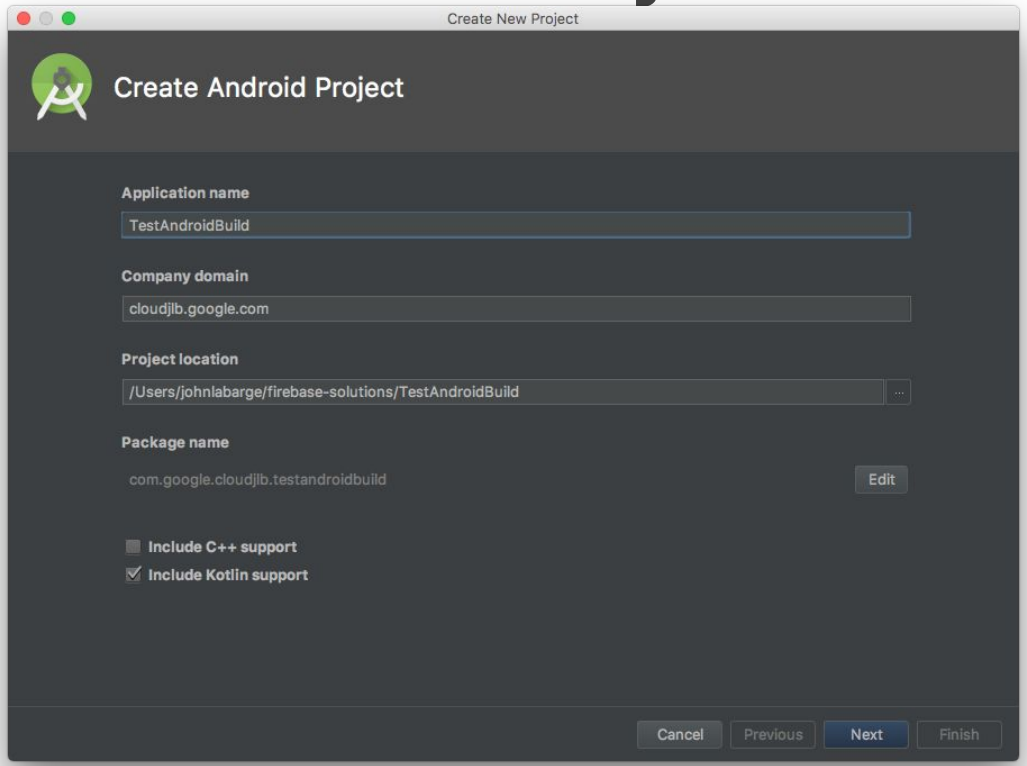
Google Cloud

Create and configure GCP project

```
export PROJECT=cloudjlb-testslides

gcloud projects create $PROJECT --organization=$GCP_ORG

gcloud beta billing projects link $PROJECT --billing-account=$(gcloud beta billing
accounts list | grep $GCP_BILLING | awk '{print $1}')

gcloud config configurations create velocity18

gcloud config set project $PROJECT
```

Google Cloud

Create an Android Project



Google Cloud

Git a repository

```
cd <project-directory>
git init
curl https://raw.githubusercontent.com/github/gitignore/master/Android.gitignore
-o .gitignore

cat >> .gitignore <<EOF
*.jks
fabric.properties
keystore.properties
EOF

git add -A
git commit -m "empty application"
```

Git remote

```
gcloud source repos create android-application

git config credential.'https://source.developers.google.com'.helper gcloud.sh

git remote add google \
    https://source.developers.google.com/p/${PROJECT}/r/android-application
```

Create a keystore

```
JKS_PASSWORD=$(openssl rand -base64 12)
keytool -genkey -noprompt -keystore android.jks \
  -alias android-key \
  -dname "CN=example.com, OU=IT, O=Example, L=Sunnyvale, S=California, C=US" \
  -storepass ${JKS_PASSWORD} \
  -keypass ${JKS_PASSWORD}

 cat > keystore.properties <<EOF
 storeFile=../android.jks
 storePassword=${JKS_PASSWORD}
 keyPassword=${JKS_PASSWORD}
 keyAlias=android-key
 EOF
```

Google Cloud

Update gradle build

```
 def betaKeystorePropertiesFile = rootProject.file("keystore.properties")
 def keystoreProperties = new Properties()
 keystoreProperties.load(new FileInputStream(betaKeystorePropertiesFile));


android {
```

Google Cloud

Update gradle build

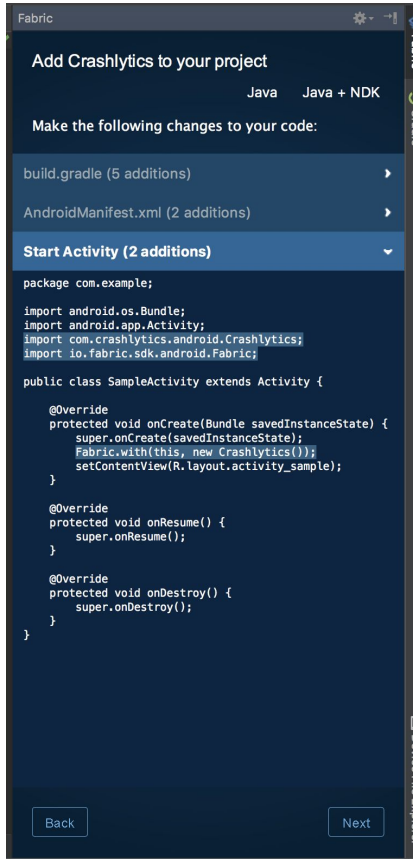
```
android {
...
   defaultConfig {...}
   signingConfigs {
       beta {
           keyAlias keystoreProperties['keyAlias']
           keyPassword keystoreProperties['keyPassword']
           storeFile file(keystoreProperties['storeFile'])
           storePassword keystoreProperties['storePassword']
       }
   }
```

Update gradle build

```
buildTypes {
       release {...}
       beta {
         initWith release
         signingConfig signingConfigs.beta
         ext.betaDistributionGroupAliases="testers"
         minifyEnabled false
         proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
       }
}
```

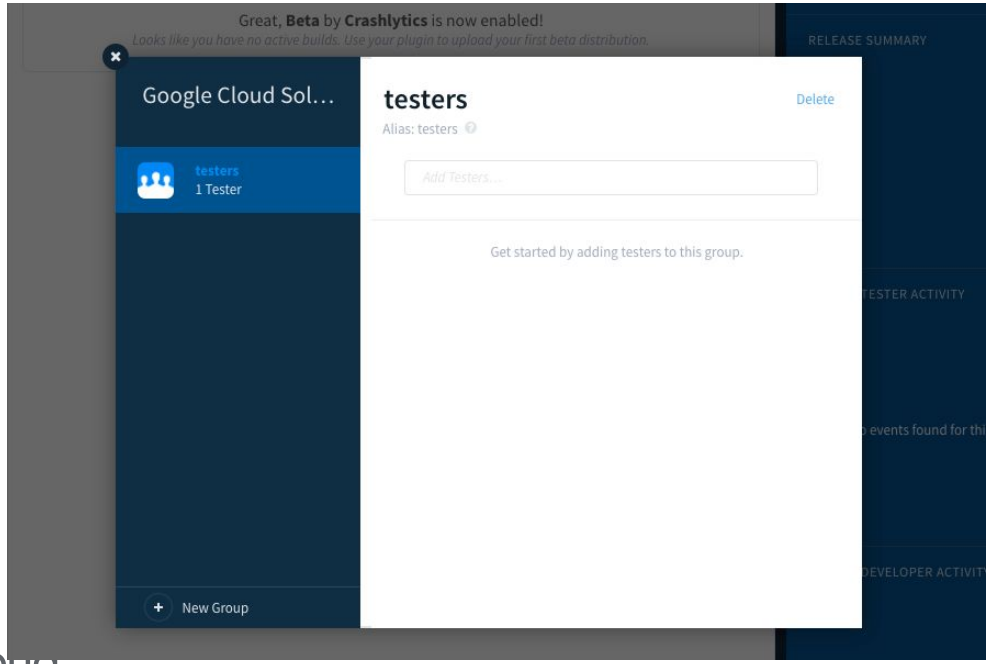
Google Cloud

Install Fabric Crashlytics



Google Cloud

Create a beta distribution group



Google Cloud

Make sure it’s working locally

```
gradle assembleBeta crashlyticsUploadDistributionBeta
```

Prepare the Android Builder

```
export ANDROID_SIGNING_BUCKET=velocity18-signing

export FABRIC_KEYS_BUCKET=velocity18-fabric

export ANDROID_BUILD_CACHE=velocity18-buildcache

gsutil mb gs://$ANDROID_SIGNING_BUCKET

gsutil cp *.jks gs://$ANDROID_SIGNING_BUCKET

gsutil mb gs://$FABRIC_KEYS_BUCKET

gsutil cp app/fabric.properties gs://$FABRIC_KEYS_BUCKET

gsutil mb gs://$ANDROID_BUILD_CACHE
```

Google Cloud

Create the builder docker image

```
git clone https://github.com/GoogleCloudPlatform/cloud-builders-community

cd cloud-builders-community/android

./installed-package-list.sh

### EDIT packages.txt to remove non-essential build packages

ANDROID_SDK_LICENSE=$(tail -1 $ANDROID_SDK/licenses/android-sdk-license)

gcloud container builds submit --config cloudbuild.yaml . \
    --substitutions=_ANDROID_SDK_LICENSE=$ANDROID_SDK_LICENSE
```

Google Cloud

Configure Android Build

```
cp ${HOME}/cloud-builders-community/android
android-cloud-build-example/fabric-beta-dist-cloudbuild.yaml ./cloudbuild.yaml

#EDIT cloudbuild.yaml

git push google master
```

Google Cloud

Configure Android Build Trigger

`_SIGNING_BUCKET_PATH`	The URL of the bucket that you created for the `.jks` and `keystore.properties`files. `gs://[ANDROID_SIGNING_BUCKET]`	Makes the Android signing credentials available to the Android build in Container Builder.
`_BUILD_TYPE`	`Beta`	Specifies the Gradle build type to use for building the application.
`_FABRIC_API_SECRET_BUCKET_PATH`	The URL of the bucket that contains the `fabric.properties` file: `gs://[FABRIC_KEYS_BUCKET]`	Provides the Fabric API credentials to the Android build.

Configure Android Build Trigger

`_EMAIL_DISTRIBUTION_GROUP`	The name of the email distribution group that you created within the Fabric UI for distributing beta versions.	Selects a list of email addresses to which Fabric will distribute the beta version of the application.
`_ANDROID_BUILD_CACHE`	The name of a Cloud Storage bucket that you use to store Gradle dependencies between builds.	Prevents Gradle from downloading the same dependencies from the internet with each build.
`_ANDROID_SDK_LICENSE`	Your Android SDK license	Verifies that you've accepted the terms of the Android SDK license.

Configure Android Build

```
cp ${HOME}/cloud-builders-community/android
android-cloud-build-example/fabric-beta-dist-cloudbuild.yaml ./cloudbuild.yaml

#EDIT cloudbuild.yaml

git push google master
```

Google Cloud

Firebase Test Lab

Generate detailed reports and screenshots to help identify bugs before you launch your app to users.

Run custom test scripts on hundred of device configurations across common Android devices.

Supplement your existing workflow through integration with Android Studio, command-line tools, and Web-based consoles.

TODO : Add a Smoke Test With Firebase Test Lab

TODO: Add a Functional Test

Additional Firebase Featuers

Google Cloud

Firebase Performance

Impact Capture and instrument performance metrics in production, then view insights in the Firebase console.

Traces allow you to instrument parts of your app to learn the duration of an action, then attach custom metrics to that action.

Network activity monitoring gives you information about response time, payload size, and success rate for HTTP/S requests.

Google Analytics

for Firebase

Measure up to 500 different event types and 25 user properties per event. Capture over 15 events automatically, including app uninstall.

Measure performance for ad campaigns on over 50 networks.

Stream Analytics events to BigQuery for full ad-hoc analytics and customized reporting workflows. Export to any 3rd party system.

Use Analytics audiences to power other features, AdWords, or trigger Cloud Functions.

Takeaways

- GCP and Firebase are working to make devops simpler for mobile developers
- Container Builder **CAN** build your Android app
- Crashlytics Beta **distributes** your Android app
- You can hook these technologies together for a simple delivery pipeline
- Firebase Test Lab is available for functional and robo testing of your Android and iOS apps!
- Monitor your apps performance and user happiness using Firebase Performance, Google Analytics for Firebase and Crash Reporting (through Firebase Crash Reporting or Crashlytics)

That’s a wrap.

Google Cloud


NEXT
18
g.co/next18
July 24-27, 2018
San Francisco