Automating Deployments with CI & CD Using BitBucket Pipelines

Maintained by

Eric Jiang

This presentation's code/slides can be found on
https://github.com/lorderikir/cicd-techtalk

Bitbucket Google Cloud

Slide Deck Version: v1.0.0

Memes Warning: You might see many memes in this talk

So what is this talk about?

1. Introduction to BitBucket Pipelines
2. Branch Workflow and Setting up Permissions
3. Automating Testing with Pipelines
4. Automating Deployments to Google App Engine
5. Building advanced pipelines

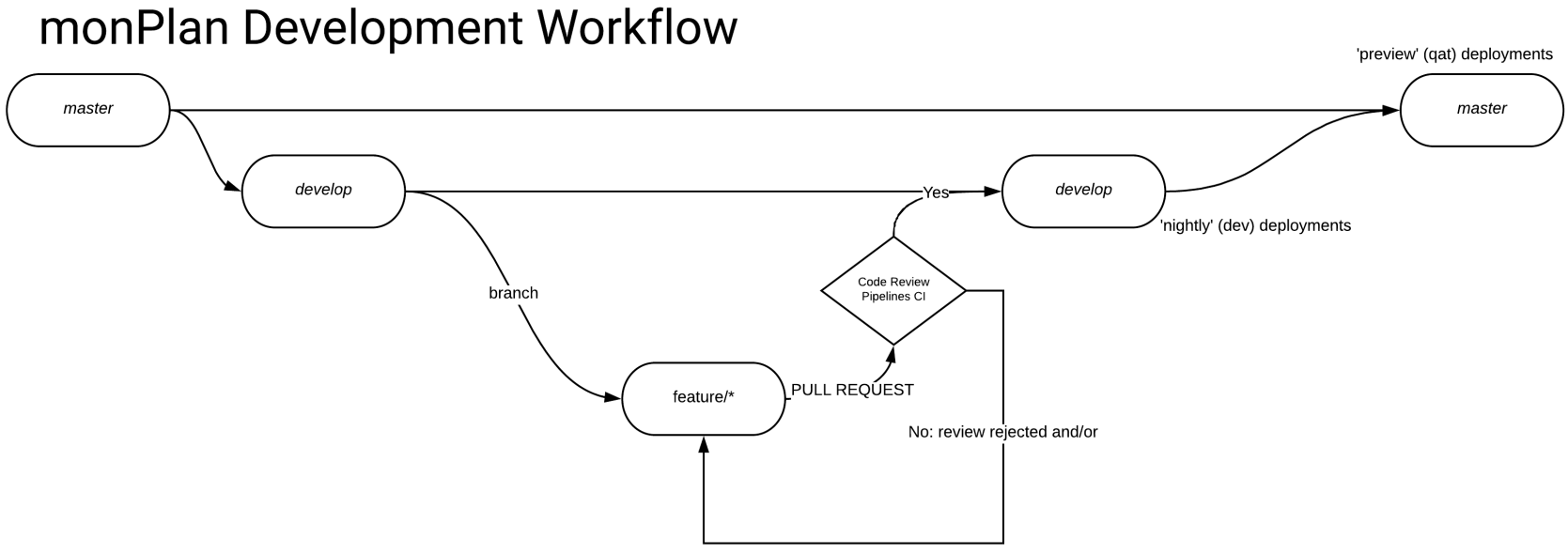
Introduction to BitBucket Pipelines

First of all, what is Pipelines?

> Integrated CI/CD for Bitbucket Cloud that's trivial to set up, automating your code from test to production.
> *BitBucket Site (Atlassian)*

So how to do it?

monPlan Git Workflow



- **master**: branch is the key branch, everytime for release
- **develop**: *unstable*, most of the PRs should go here
- **'feature/*'**, **'fix/***, etc.: are 'for purpose' branches, these branches are for development
- **deploy** (not shown), is for **manual** deployments to prod

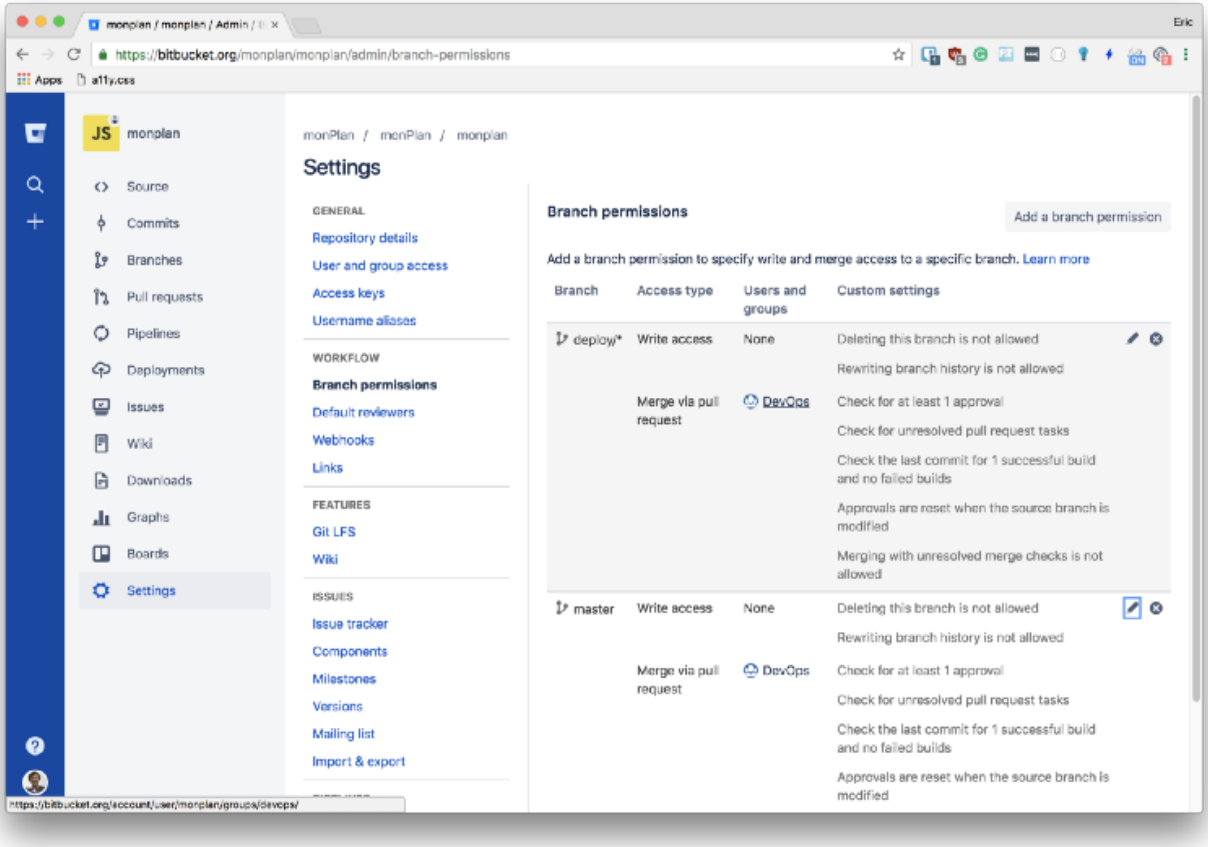
Branch Permissions

- **no-one** has write permissions to *master*, *develop* or *deploy*, not even DevOps or Admin.
- **everyone** has PR write access, only when
  - 1 Tests Pass
  - 1 Approval from a Code-review
- that way, we 'silly'-proof our codebase.

Setting Branch Permissions

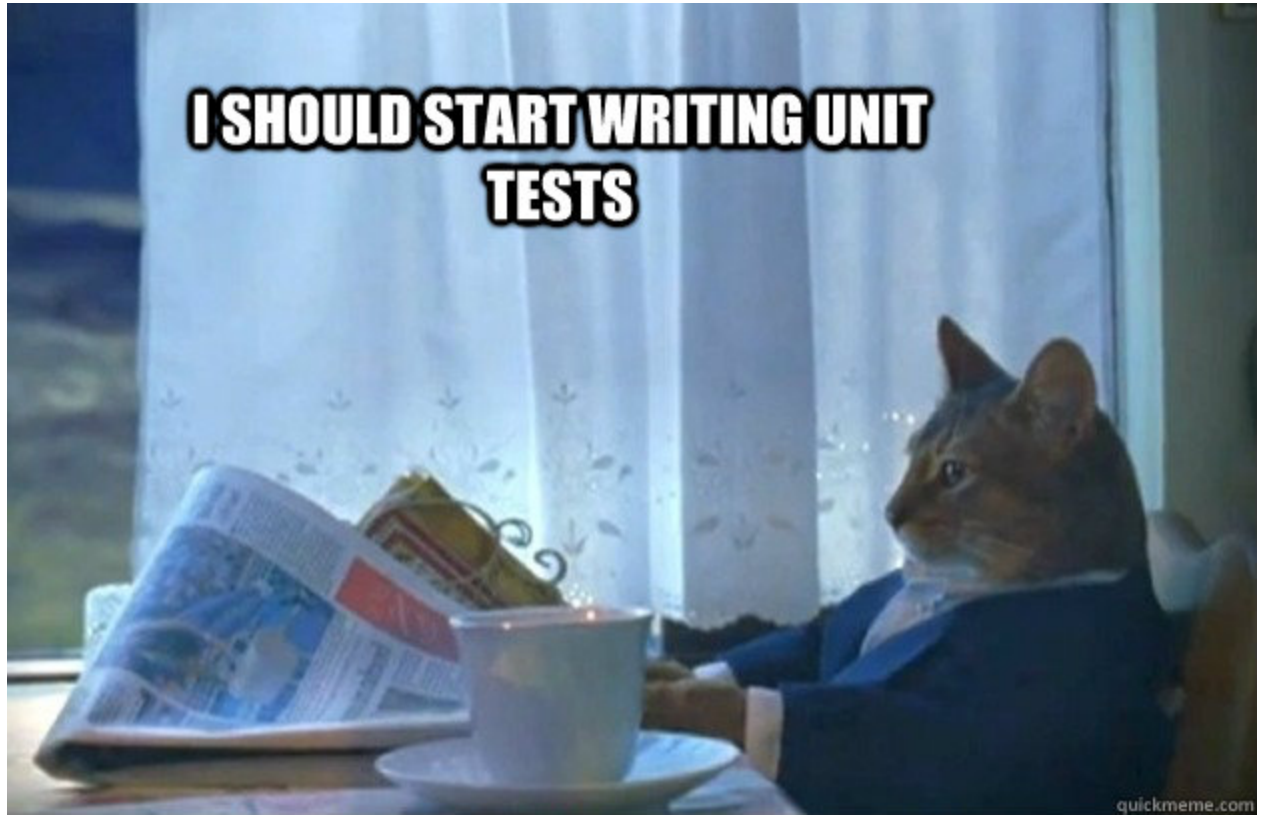
Branch permissions by people who have *administrator* privellages inside a repo.

To set this, Go to Settings and under workflow select Branch Permissions.



Demo of Branch Permissions

Automating Testing



Note that we will be using **NodeJS** for the sake of this demo/talk, but the idea is similar

First, It's the Development Team's Problem to write appropriate Tests, not the CI/CDs role

Second, the CI works on an alternative CLI, so running tests on interactive mode won't work

Setting up CI for Testing

default Pipeline runs for all branches (if not matched)

from bitbucket-pipelines.yml

```
# Use Prebuilt Node 8 Image (Node 8 supports yarn)
image: node:8


pipelines:
  default:
    - step:
        caches:
          - node
        script:
          - yarn # <- Install Dependencies
          - yarn test:ci # <- Test in Non-CI Mode
```

We can also run workflows on branches

Pattern	Targets
actual branch name	actual branch
feature/*	all branches with `feature/` such as `feature/sso`, `feature/login`, etc.
*	Matches all branches, tags, or bookmarks. The star symbol (`*`) must be in single quotes. Doesn't match those with slash
**	Matches all branches including those which have slash

So, simple we can run it with the `branches` mode

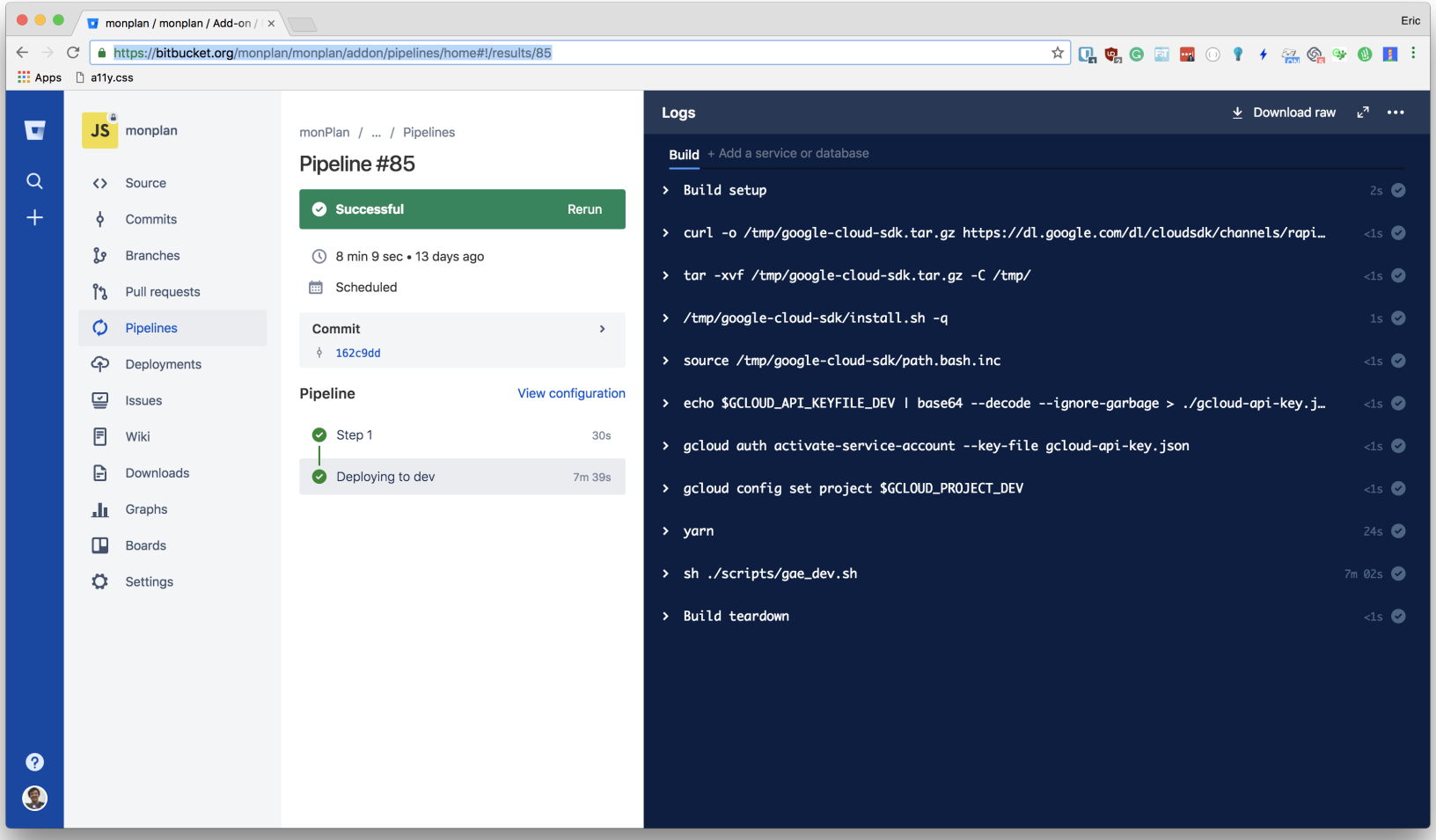
```
# Use Prebuilt Node 8 Image (Node 8 supports yarn)
image: node:8


pipelines:
  default:
    - step:
        caches:
          - node
        script:
          - yarn # <- Install Dependencies
          - yarn test:ci # <- Test in Non-CI Mode
  branches:
        # run branches here
```

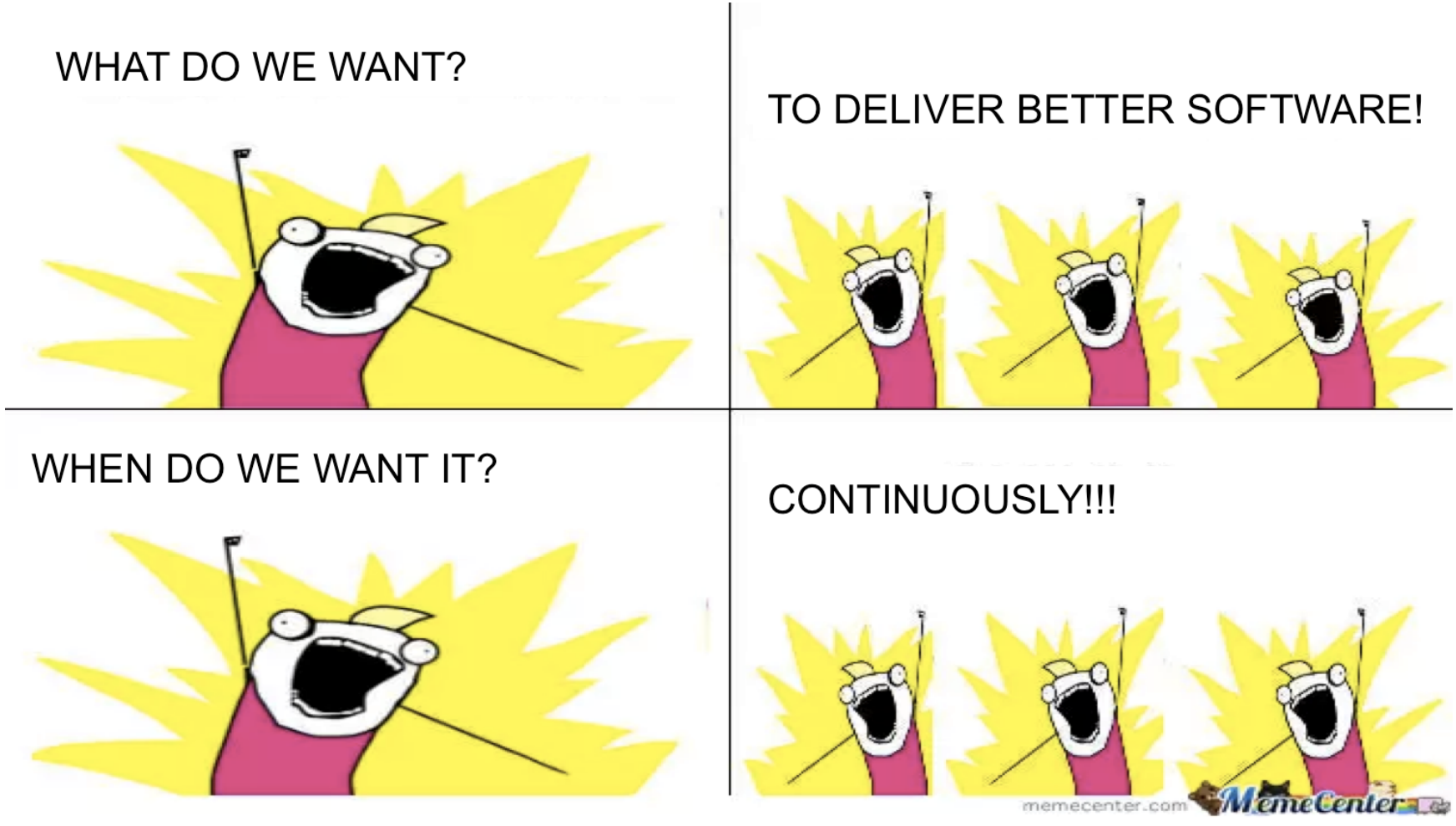
What if we want to have testing on certain branches, such as `develop`.

```
# .... (reduced for verbosity)
  branches:
        - step:
          name: Run Unit Tests # <- We can name steps
          caches:
            - node
          script:
            - yarn # <- Install Dependencies
            - yarn test:ci # <- Test in Non-CI Mode
```

An example of a pipeline (incl. deployments)



Now, we can automate deployments.



Automating Deployments to Google App Engine

We can automate most of delivery.

1. Create Service Account with Details, Create Private Key and Download as JSON
2. Encode JSON under base64 and upload as an environment variable in repo settings
3. Pipeline deployment script will need to install Google Cloud SDK first
4. (Build site - ReactJS) and deploy using `google app deploy`