Concourse:

CI that scales with your project

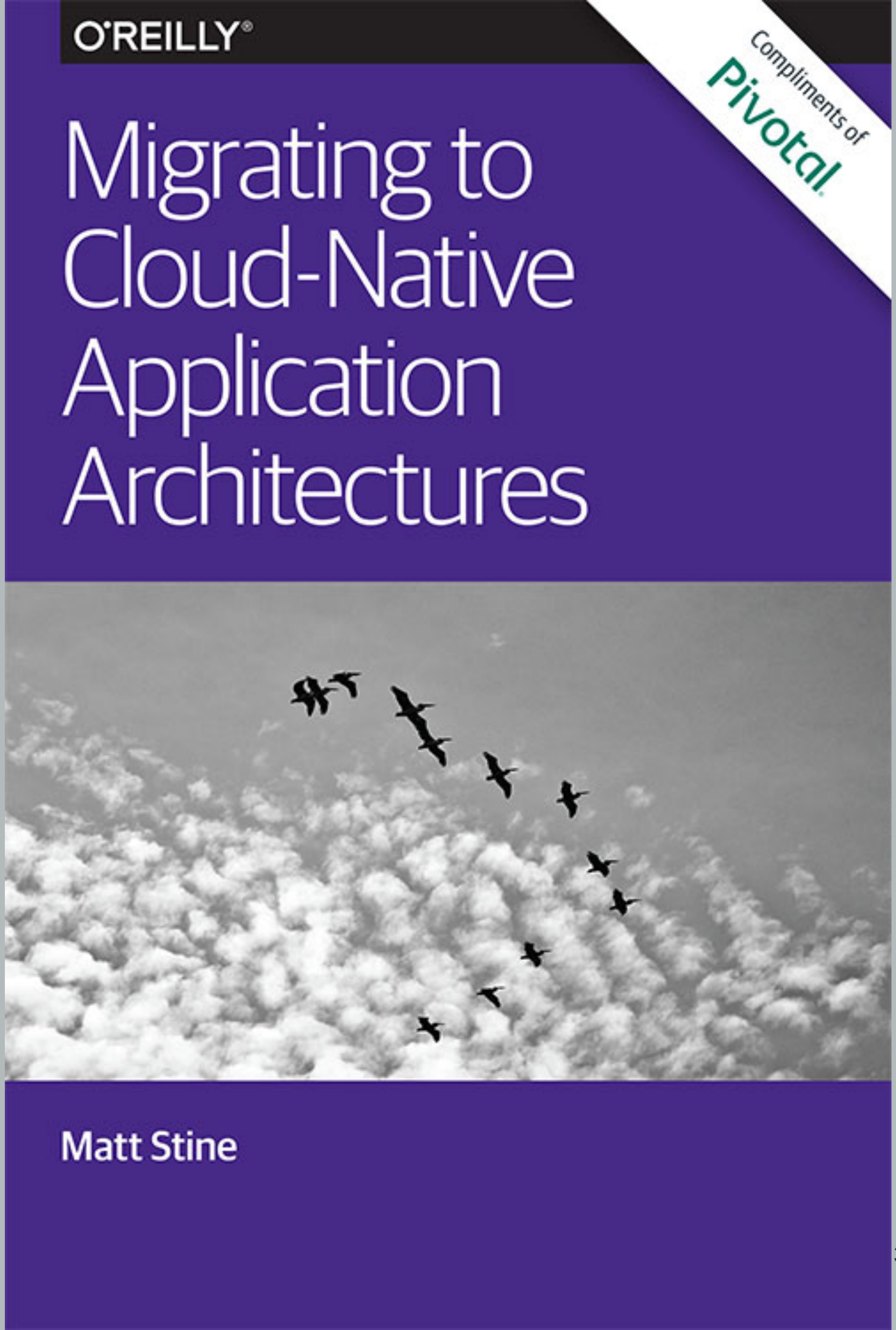
Me

Matt Stine @mstine
Strategic Product Owner - Spring Portfolio
Pivotal Software, Inc.
matt.stine@gmail.com

I wrote a little cloud book...

FREE - Compliments of Pivotal

http://bit.ly/cloud-native-book



What is Concourse?

- Open Source CI Pipeline system
- Developed by Pivotal
- http://concourse.ci

Because the world needed another CI system…

0_o

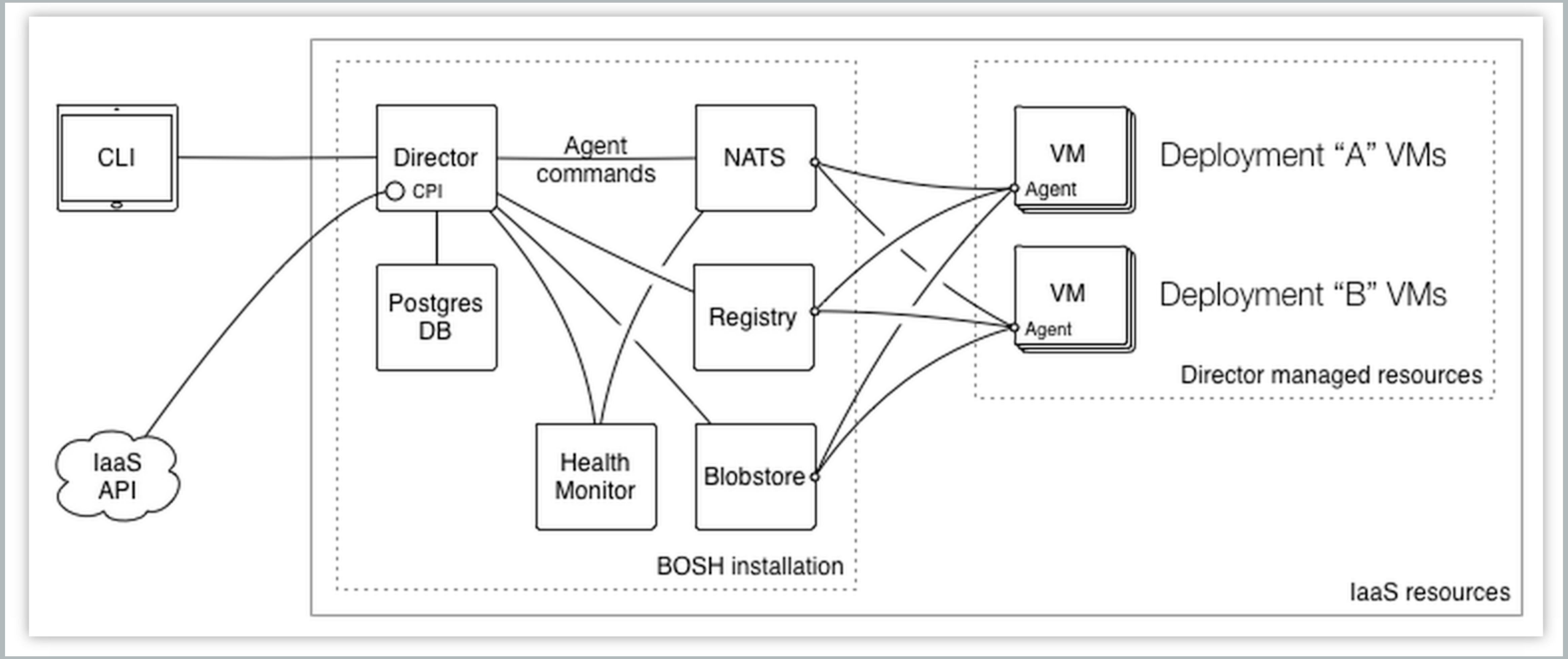
Why?

Simplicity

Usability

Build Isolation

Scalable/Reproducible deployment



http://bosh.io

Flexibility

Local Iteration

Concepts

Running Example:

- Consumer-Driven Contract Testing (http://martinfowler.com/articles/consumerDrivenContracts.html)
- Using Pact-JVM (https://github.com/DiUS/pact-jvm)
- Example Project (https://github.com/mstine/microservices-pact)

Tasks

execution of a script in an isolated environment with dependent resources made available to it

```
---
platform: linux
image: docker:///java#8
inputs:
- name: microservices-pact
- name: foo-consumer-version
outputs:
- name: pacts
- name: libs
params:
    TERM: dumb
    VERSION_FILE_PATH: foo-consumer-version
run:
  path: microservices-pact/gradlew
  args:
  - -b
  - microservices-pact/build.gradle
  - :microservices-pact-consumer:test
  - :microservices-pact-consumer:assemble
```

```
platform: linux
image: docker:///java#8
inputs:
- name: microservices-pact
- name: pact
- name: foo-provider-version
outputs:
- name: provider-libs
params:
  TERM: dumb
  PACT_FILE: ../pact/Foo_Consumer-Foo_Provider.json
  VERSION_FILE_PATH: foo-provider-version
run:
   path: microservices-pact/gradlew
   args:
   - -b
   - microservices-pact/build.gradle
   - :microservices-pact-provider:assemble
   - :microservices-pact-provider:pactVerify
```

Resources

data: inputs/outputs

Can be...

- Checked
- Fetched
- Pushed

Git

```
- name: microservices-pact
  type: git
  source:
    uri: https://github.com/mstine/microservices-pact.git
    branch: master
```

S3 Bucket

```
- name: foo-consumer
  type: s3
  source:
    access_key_id: {{access_key_id}}
    secret_access_key: {{secret_access_key}}
    bucket: concourse-pact
    regexp: microservices-pact-consumer-(.*).jar$
```

Semantic Versioning

```
- name: foo-consumer-version
  type: semver
  source:
    bucket: concourse-pact
    key: foo-consumer-version
    access_key_id: {{access_key_id}}
    secret_access_key: {{secret_access_key}}
    initial_version: 0.1.0
```

Cloud Foundry!

```
- name: pws-deploy
  type: cf
  source:
    api: https://api.run.pivotal.io
    username: {{pws_username}}
    password: {{pws_password}}
    organization: platform-eng
    space: concourse-demo
    skip_cert_check: false
```

Built-In Resources
http://concourse.ci/resource-types.html

- The git resource can pull and push to git repositories.
- The time resource can start jobs on a schedule or timestamp outputs.
- The s3 resource can fetch from and upload to S3 buckets.
- The archive resource can fetch and extract .tar.gz archives.
- The semver resource can set or bump version numbers.
- The github-release resource can fetch and publish versioned GitHub resources.
- The docker-image resource can fetch, build, and push Docker images
- The tracker resource can deliver stories and bugs on Pivotal Tracker
- The pool resource allows you to configure how to serialize use of an external system. This lets you prevent test interference or overwork on shared systems.
- The cf resource can deploy an application to Cloud Foundry.
- The bosh-io-release resource can track and fetch new BOSH releases from bosh.io.
- The bosh-io-stemcell resource can track and fetch new BOSH stemcells from bosh.io.
- The bosh-deployment resource can deploy BOSH stemcells and releases.
- The vagrant-cloud resource can fetch and publish Vagrant boxes to Atlas.

Growing List of Community Resources, including:

http://concourse.ci/resource-types.html

- Slack
- Pull Requests
- Email
- Bintray
- Perforce
- FTP
- Twitter
- HipChat
- Bitbucket
- Terraform
- Rsync
- JIRA
- Google Drive

Implement Your Own

- Docker Image w/ 3 Scripts
- `/opt/resource/check`
- `/opt/resource/in`
- `/opt/resource/out`
- Add to your Concourse deploy via `resource_types` section in pipeline config:

```
resource_types:
- name: pivnet
  type: docker-image
  source:
    repository: pivotalcf/pivnet-resource
    tag: latest-final
```

- http://concourse.ci/implementing-resources.html

Jobs

functions composed of behavior (tasks) and inputs/outputs (resources/other jobs)

Jobs Have Builds

- Success (all tasks succeed)
- Failure (any task fails)
- Can be accessed while running/shortly after finish (intercept/hijack)

Jobs Have Plans

- Sequence of steps to execute:
- *get* resources
- run things (*task*)
- *put* resources
- parallel or serial

Verify Pact (inputs)

```
- get: microservices-pact
  passed: [generate-pact]
  trigger: true
- get: foo-provider-version
  params: {bump: minor, pre: alpha}
- get: pact
  passed: [generate-pact]
  trigger: true
```

Verify Pact (function)

```
- task: verify-pact
  file: microservices-pact/microservices-pact-provider/task.yml
```

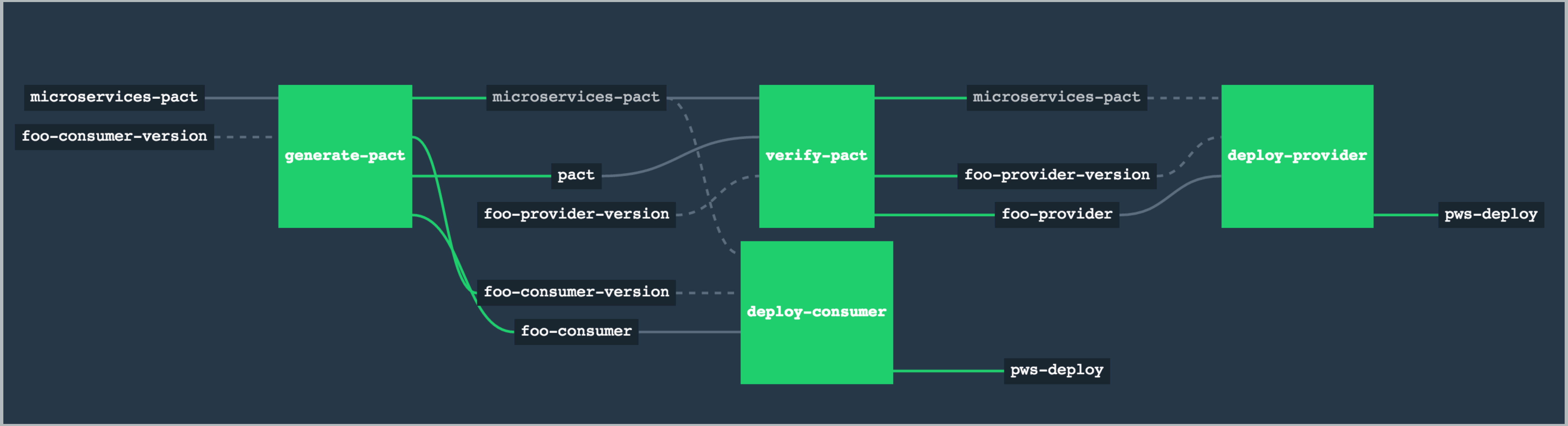
Verify Pact (task)

```
platform: linux
image: docker:///java#8
inputs:
- name: microservices-pact
- name: pact
- name: foo-provider-version
outputs:
- name: provider-libs
params:
  TERM: dumb
  PACT_FILE: ../pact/Foo_Consumer-Foo_Provider.json
  VERSION_FILE_PATH: foo-provider-version
run:
   path: microservices-pact/gradlew
   args:
   - -b
   - microservices-pact/build.gradle
   - :microservices-pact-provider:assemble
   - :microservices-pact-provider:pactVerify
```

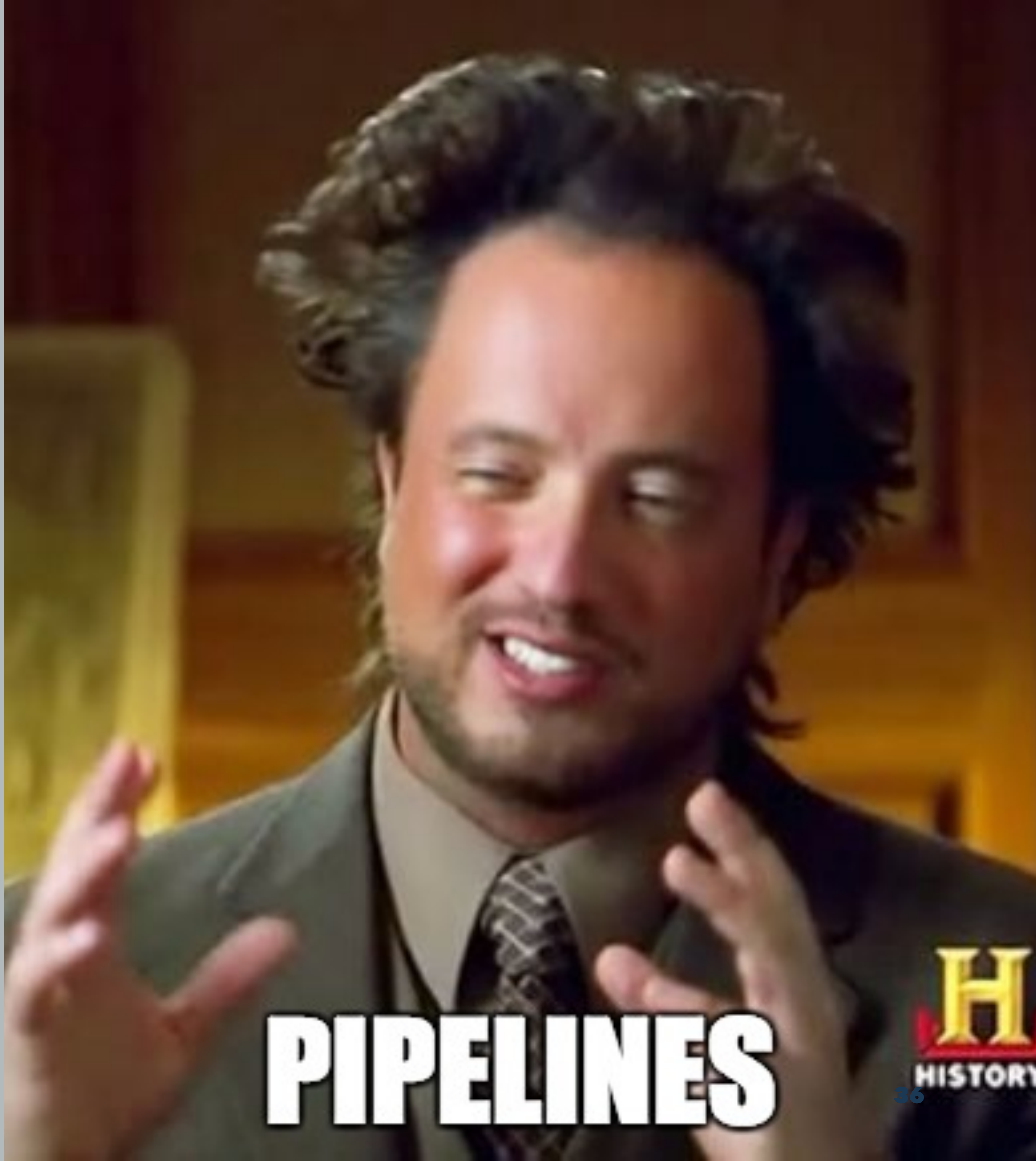
Verify Pact (outputs)

```
- put: foo-provider
  params: {file: provider-libs/microservices-pact-provider-*.jar}
- put: foo-provider-version
  params: {file: foo-provider-version/number}
```

Pipelines


microservices-pact
foo-consumer-version
generate-pact
microservices-pact
pact
foo-provider-version
foo-consumer-version
foo-consumer
verify-pact
deploy-consumer
microservices-pact
foo-provider-version
foo-provider
pws-deploy
deploy-provider
pws-deploy

That's It

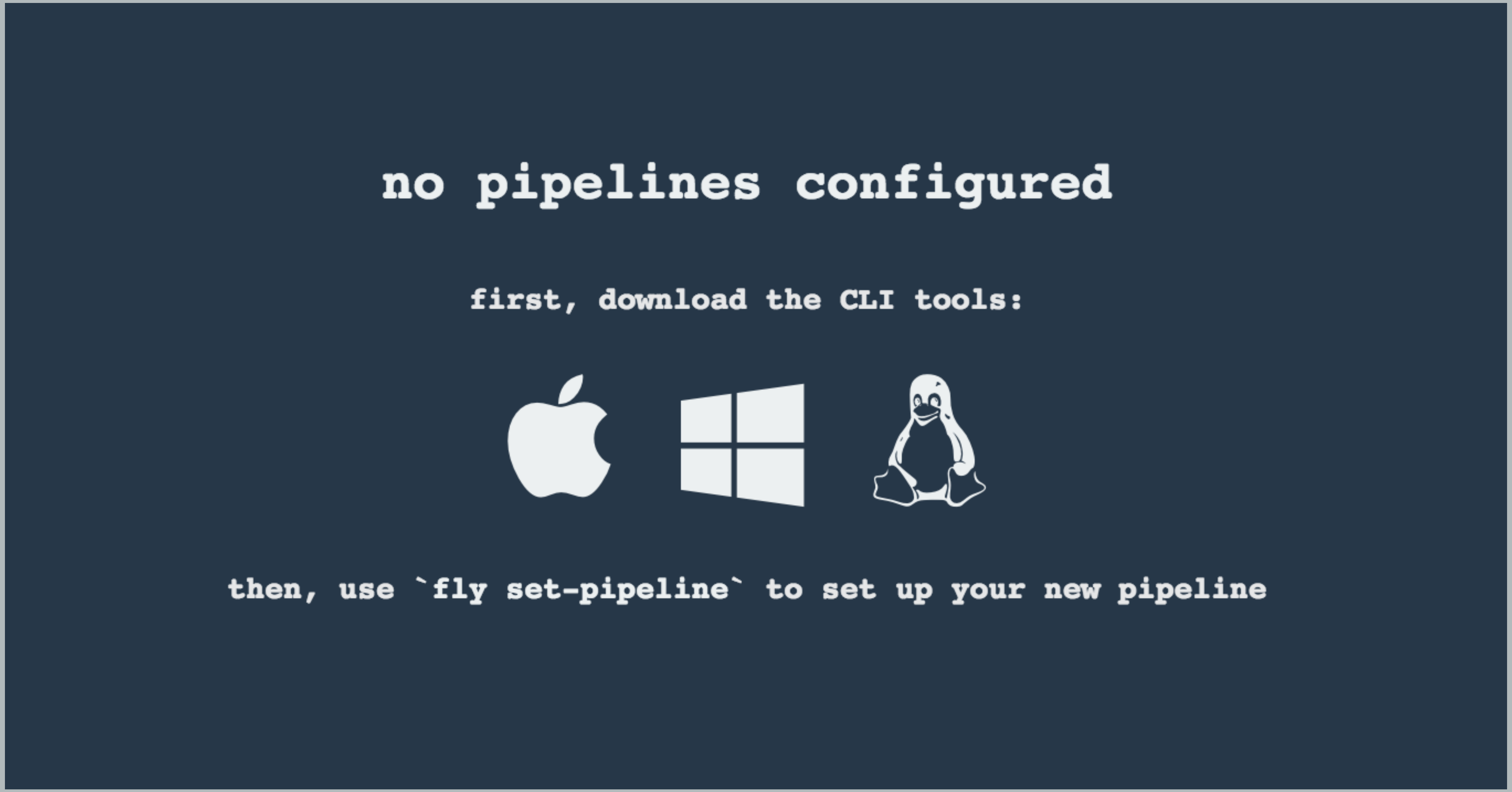


Learning to Fly

Getting Started

```
$ vagrant init concourse/lite
$ vagrant up
```

http://192.168.100.4:8080



no pipelines configured

first, download the CLI tools:

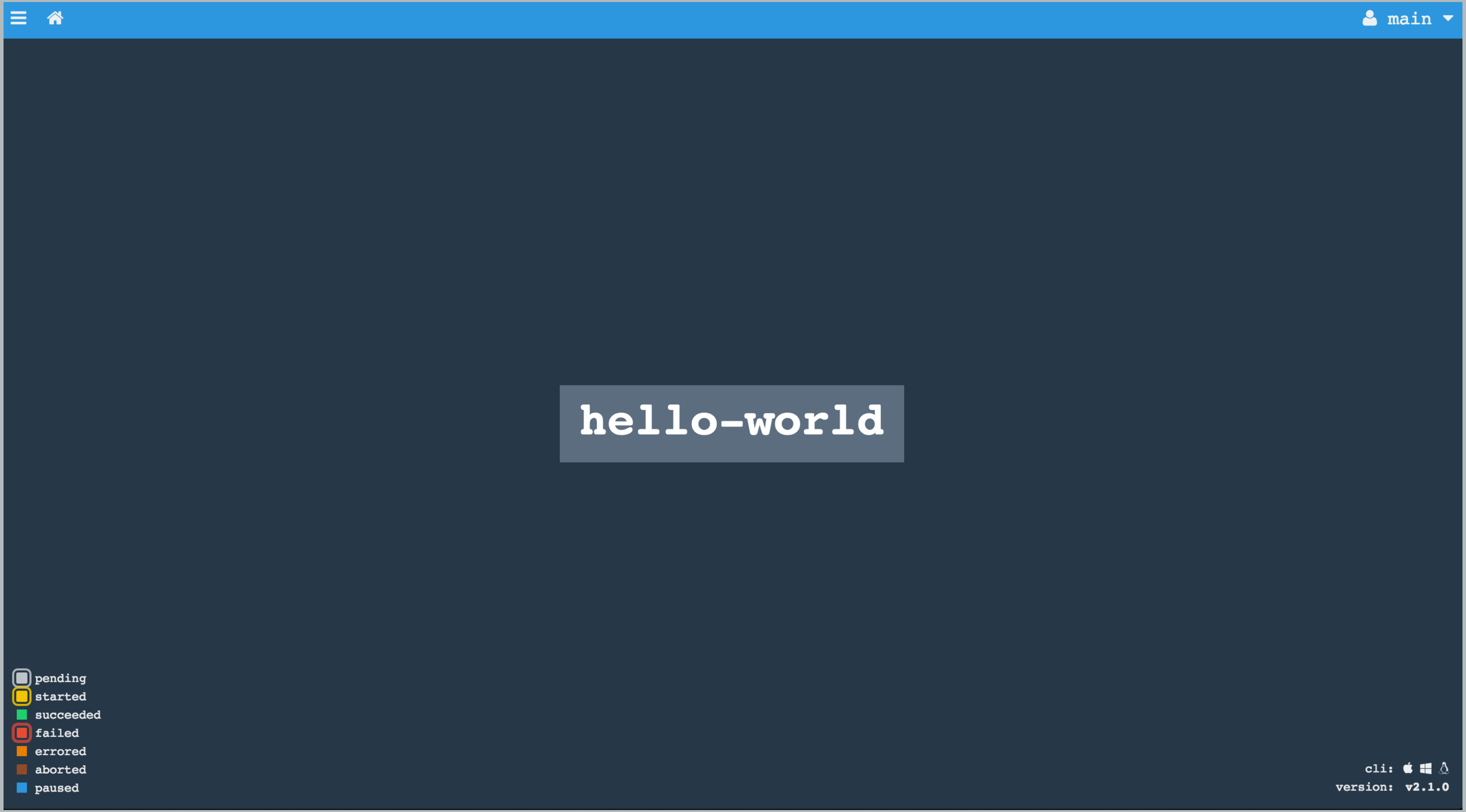
then, use `fly set-pipeline` to set up your new pipeline

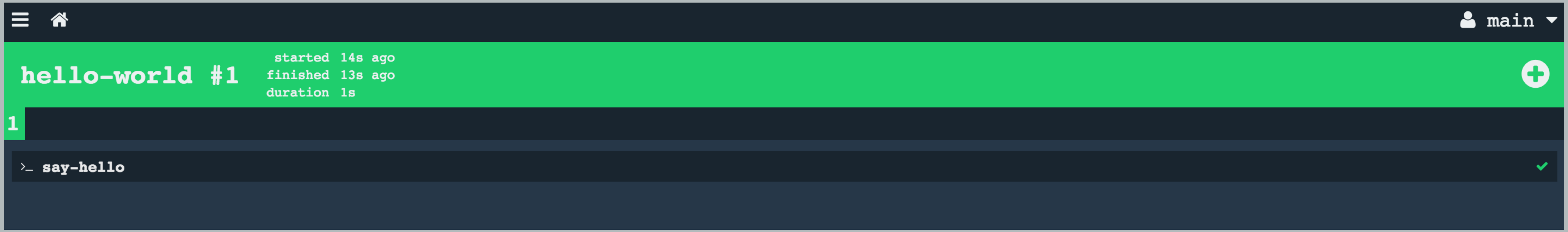
Let's do this...

```yaml
jobs:
- name: hello-world
  plan:
  - task: say-hello
    config:
      platform: linux
      image_resource:
        type: docker-image
        source:
          repository: busybox
      run:
        path: echo
        args: ["Hello, World!"]
```

Ship It!

```
$ fly set-pipeline -p hello-world -c hello-world-pipeline.yml
```


main
hello-world
pending
started
succeeded
failed
errored
aborted
paused
cli:
version: v2.1.0



```
hello-world #1   started  14s ago
                 finished 13s ago
                 duration 1s
1
>_ say-hello ✓
```

Let's Play

Intercepting with Fly

Intercepting a Job Step

```
fly -t <target> intercept -j <pipeline>/<job> -b <build #> -
s <step>
```

Intercepting a Resource

```
fly -t <target> intercept --check <pipeline>/<resource> /
bin/sh
```

Thanks!

Matt Stine (@mstine)

- *This Presentation:* https://github.com/mstine/nfjs_2015/tree/master/Concourse
- *Example Project:* https://github.com/mstine/microservices-pact
- *Concourse Website:* http://concourse.ci
- *Concourse Slack Team:* https://concourseci.slack.com