



# Concourse:

CI that scales with your project



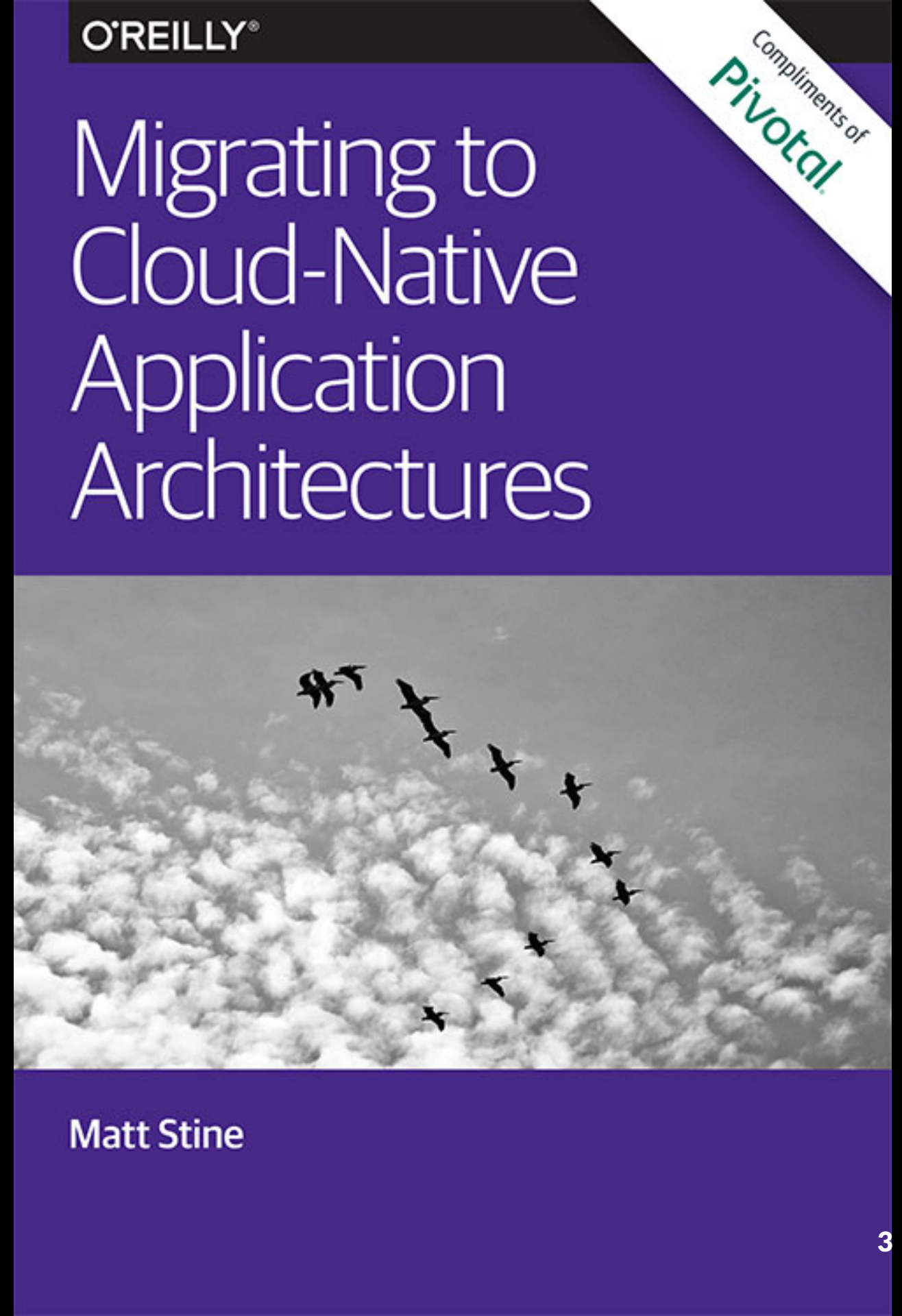
# Me

Matt Stine [@mstine](#)  
Senior Product Manager  
Pivotal  
[matt.stine@gmail.com](mailto:matt.stine@gmail.com)

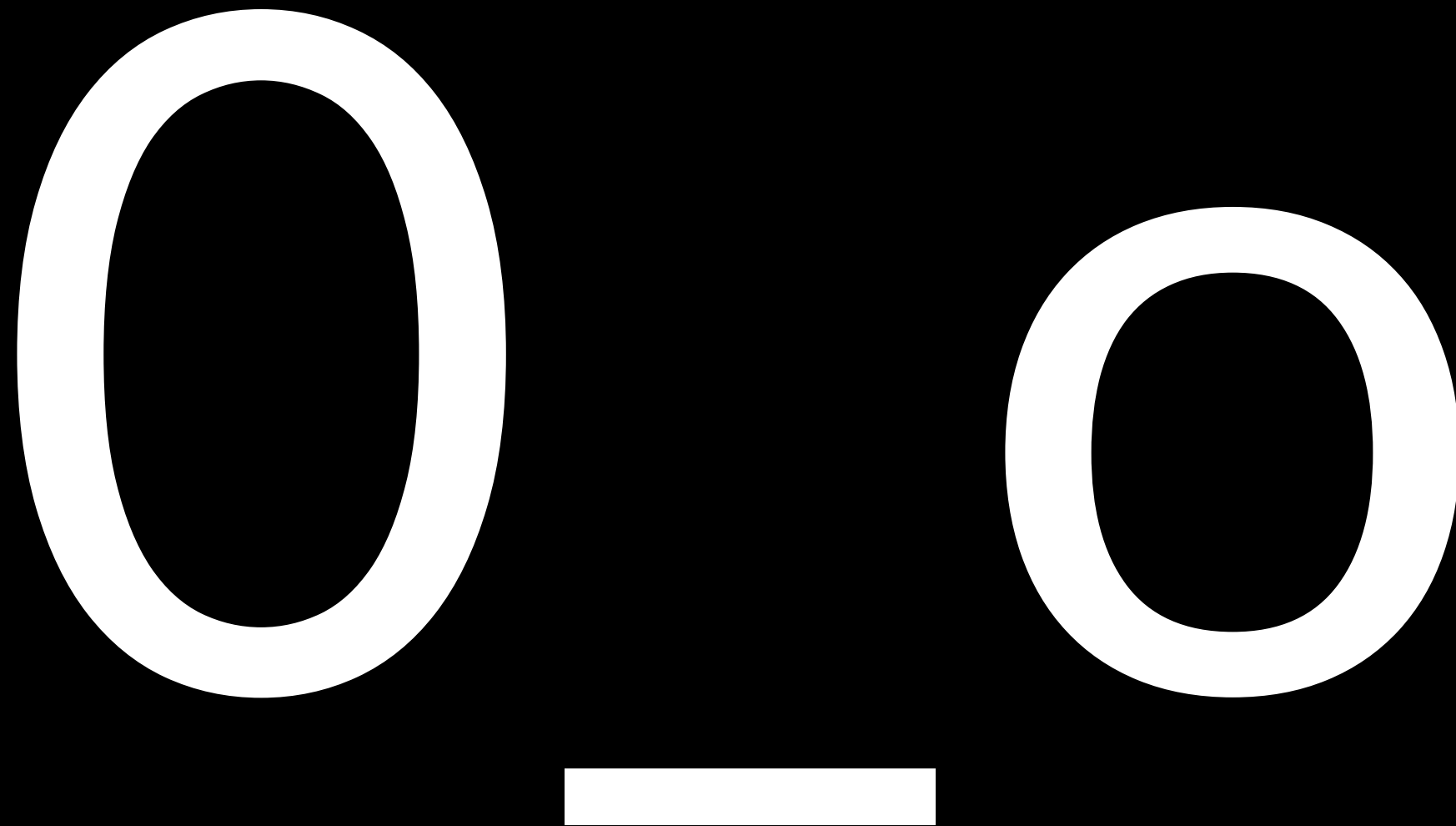
# I wrote a little cloud book...

FREE - Compliments of Pivotal

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



Because the world needed another  
CI system...



Why?

# Simplicity

# Usability

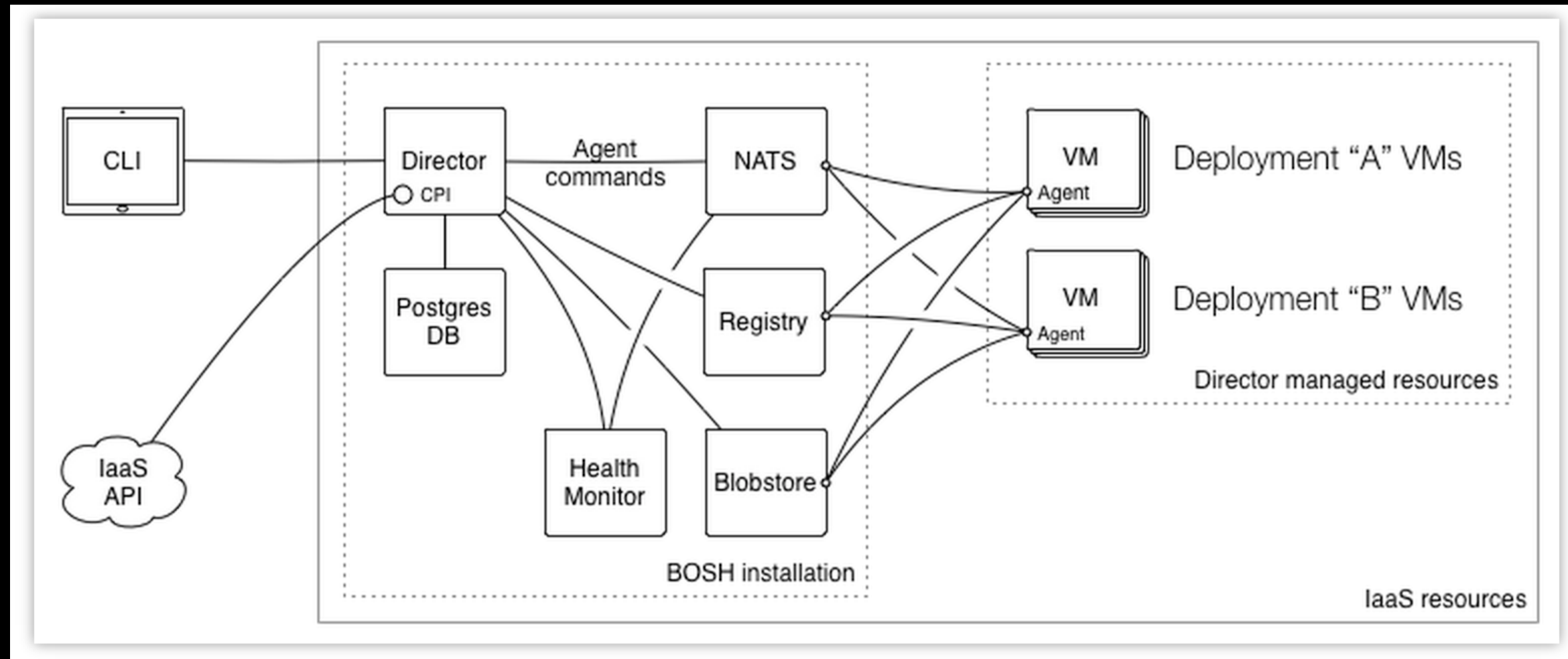




# Build Isolation



# 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  
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
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
```



# <https://github.com/concourse?query=resource>

- pool
- git
- vagrant-cloud
- docker-image
- cf
- s3
- cf service-broker
- bosh-deployment
- bosh-io-release
- bosh-io-stemcell
- pivotal tracker
- archive (tgz)
- semver
- github-release
- time

# Implement Your Own

<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)

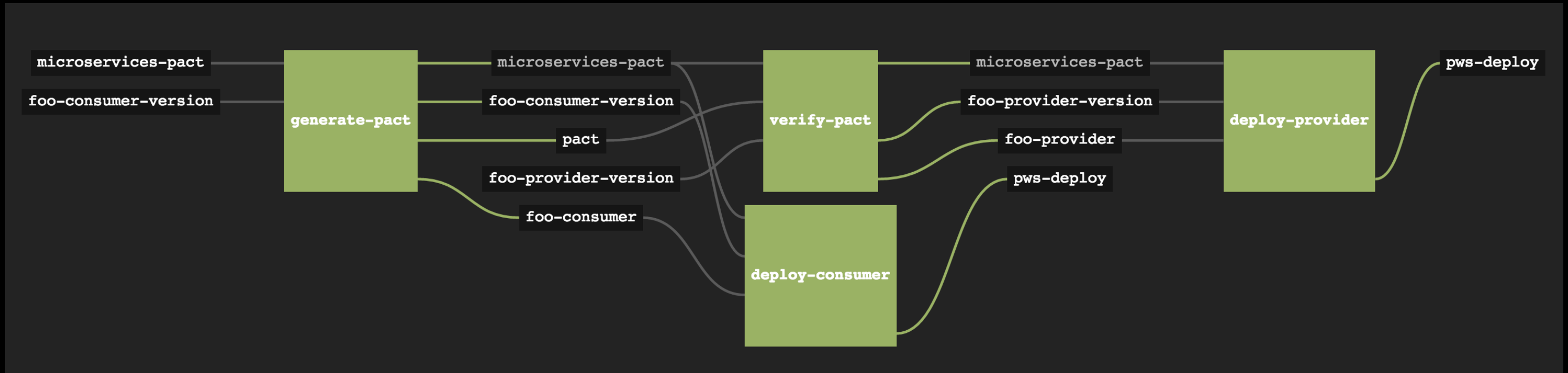
```
platform: linux
image: docker:///java#8
inputs:
- name: microservices-pact
- name: pact
- name: foo-provider-version
params:
  TERM: dumb
  PACT_FILE: ../pact/Foo_Consumer-Foo_Provider.json
  VERSION_FILE_PATH: foo-provider-version
run:
  path: microservices-pact/gradlew
  args:
  - -b
  - microservices-pact/build.gradle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

# Verify Pact (outputs)

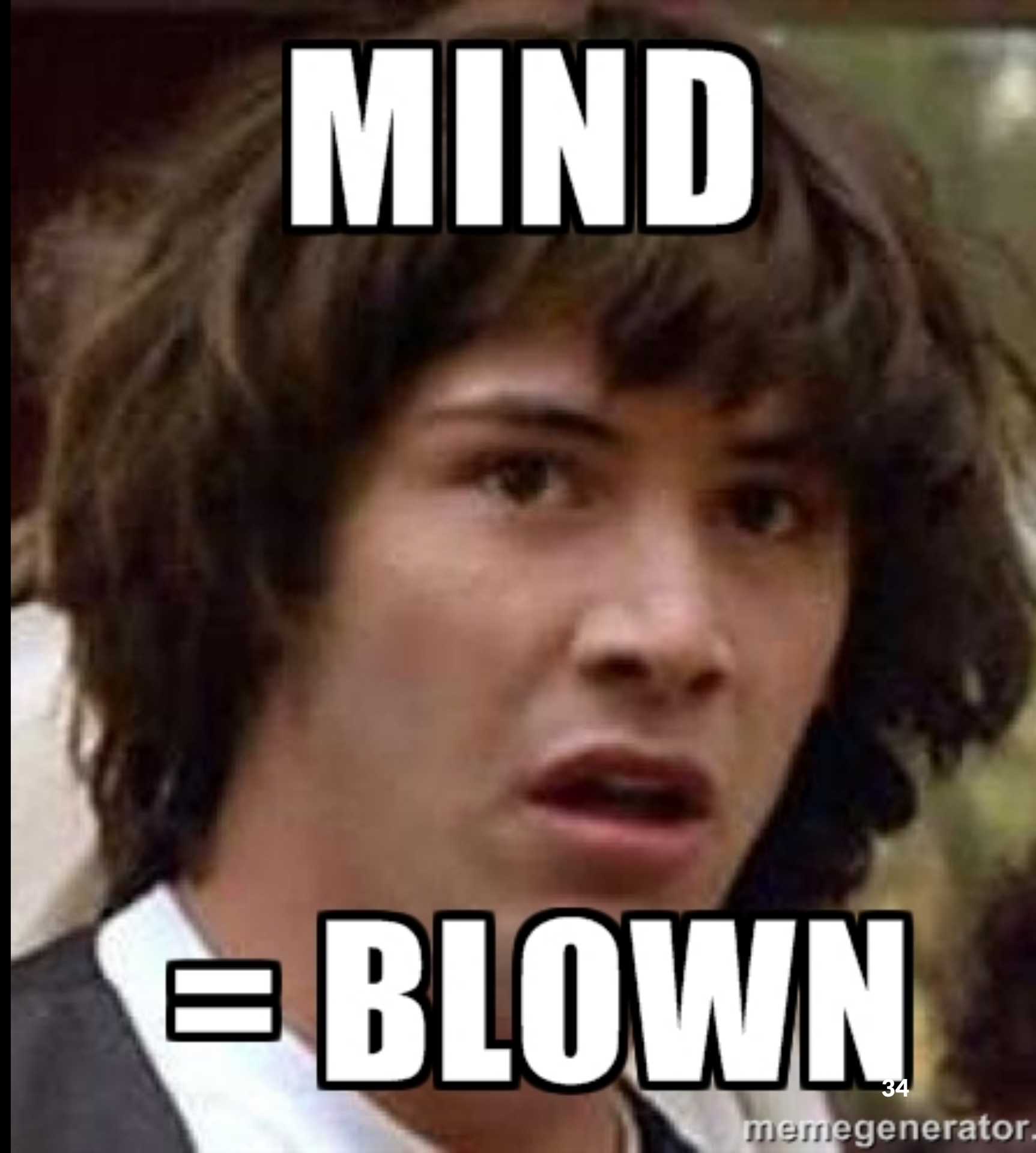
- put: foo-provider  
params: {from: microservices-pact-provider/build/libs/microservices-pact-provider-\*.jar\$}
- put: foo-provider-version  
params: {file: foo-provider-version/number}

# Pipelines





That's  
It





# Learning to Fly

# Getting Started

```
$ git clone https://github.com/concourse/concourse.git  
$ vagrant init concourse/lite  
$ vagrant up
```



<http://192.168.100.4:8080>

**no pipelines configured**

**first, download the CLI tools:**

cli:   

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

# Let's do this...

```
jobs:  
- name: hello-world  
  plan:  
  - task: say-hello  
    config:  
      platform: linux  
      image: "docker:///busybox"  
      run:  
        path: echo  
        args: ["Hello, world!"]
```

# Ship It!

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





hello-world

 pending

 started

 succeeded




 failed

 errored

 aborted

 paused

cli:   




hello-world #1

started in a few seconds


succeeded in a few seconds

duration 9s



1

>\_ say-hello



Hello, world!



# Let's Play

# Thanks!

*Matt Stine (@mstine)*

- *This Presentation:* [https://github.com/mstine/nfjs\\_2015/tree/master/Concourse](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>