Breaking the cf-release Monolith

Alvaro Perez-Shirley & David Sabeti
A brief history of the cf-release monolith
A few advantages of breaking the monolith

- Extract and reuse components
- Find hidden dependencies & thus provide greater fault tolerance
- Establish clear contracts for components
- Test components individually & then together
- Commit code frequently & without conflicts
Continuous Integration with small teams working quickly. Or how I learned to stop worrying and love bosh releases.
Drawing lines in the sand.
Integration & Acceptance test ownership.
Integration Testing

Sole responsibility of **Release Integration team**, other teams don’t care about cloud/infrastructure config and are responsible only for maintaining their contract.

- **Team 1**
  - **Acceptance Testing**
    - Just maintain the contract & forget about config

- **Team 2**
  - **Acceptance Testing**
    - Just maintain the contract & forget about config

- **Team … n**
  - **Acceptance Testing**
    - Just maintain the contract & forget about config
Many releases & one deployment?
Leveraging bosh 2.0 Manifest Generation
BOSH 1.0

- cf-release
  - & deployment
- UAA
- Routing
- Loggregator
- CAPI
- Diego
  - (release & deployment)

BOSH 2.0

- cf-deployment
  - (config only)
- UAA
- Routing
- Loggregator
- CAPI
- Diego
What does this mean for operators & developers?

- Faster releases that are easier to compose & customize
- Simple configuration & sane defaults
- Upgrade easily, quickly & selectively
- Stable api that doesn’t change much
- Better fault tolerance
Any questions?

Thanks for attending this talk!