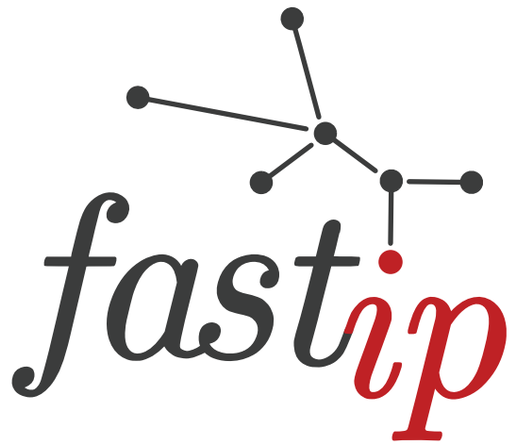
Erlang Deployments Demystified

Joe Williams (@williamsjoe)
fast_ip

Who am I?

- Engineer at fast_ip
- Background in Distributed Systems, Infrastructure and Operations
- Member of Rebar commit team
- Twitter: @williamsjoe



Context

- Erlang
- Rebar
- Chef
- Jenkins

Problem?

- Sustainable Erlang deployments are not easy
    - Very opinionated
    - Hot-code upgrades

Deployment

- Standardization
    - Directory structure
    - Config files
    - Scripts

Deployment

- Fat tarballs
  - Self contained installation
  - All the files the application needs
    - ... but nothing more
  - Dependencies may live outside

Deployment

- Process monitoring
    - Automatic restarts if process dies
    - Log rotation

Deployment

- Automation and Configuration Management
    - Consistent, repeatable deployments
    - Lives in an SCM
    - Config file templates
    - Abstraction from OS
    - Automation != Automatic

Deployment

- Workflow
    - Install dependencies (apt-get, etc)
    - Untar release
    - Adjust config files, create directories, etc
    - Start service

Deployment

- Chef
  - **NEW!** Deploy via erlang_apps
    - A simple Chef API for deploying Erlang releases

- https://github.com/fastip/fastip_cookbooks/tree/master/apps

Deployment

- Continuous Integration
    - Automated builds
        - Publish “fat tarballs” of your releases for deployment
        - Publish upgrade tarballs for hot code upgrades
- Automated testing
    - eunit, etap, eqc, dialyzer, etc

Deployment

- End to End Workflow
    - Push code
    - CI builds and publishes new code
    - Deploy?
    - Config management deploys new code
    - Profit!

Deployment

- Wait ... what about hot-code upgrades?!

Hot Upgrades

- Two steps beyond building a release
    - Generate appup files via Rebar
    - Generate upgrade tarballs via Rebar
- All done via continuous integration

Hot Upgrades

- How do they work?
  - Versioned applications and releases
  - Appup and relup files

Hot Upgrades

- We'll do it live!

Hot Upgrades

- Workflow
    - Build “old” release
    - Push new code and version numbers (or use SCM tags!)
    - Build “new” release
    - Generate appup files based on both releases
    - Generate upgrade tarball based on the appup files

Take Away

- Make deployment and upgrades easy, non-events
    - Use Rebar
    - Automate and standardize
    - Use hot code upgrades

Questions?