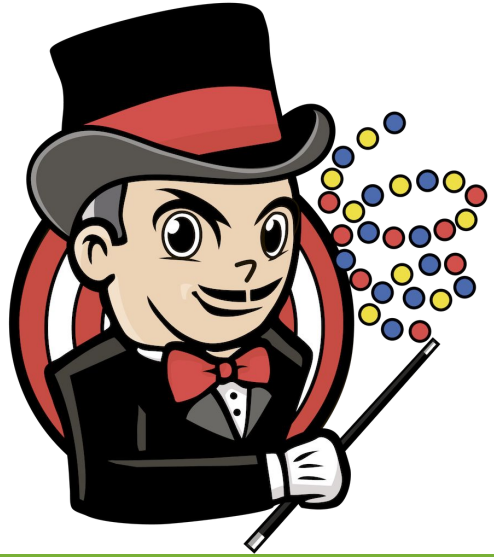
Jenkins Essentials

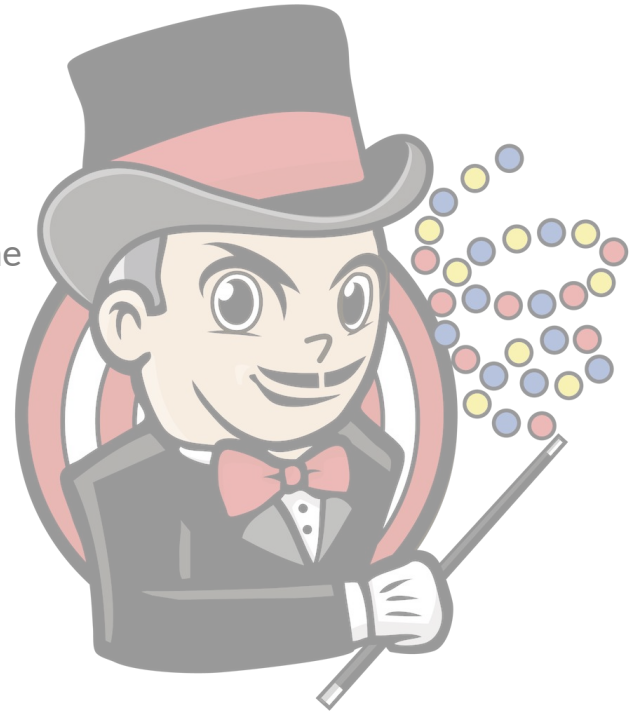
Bringing a bit of magic to Jenkins.

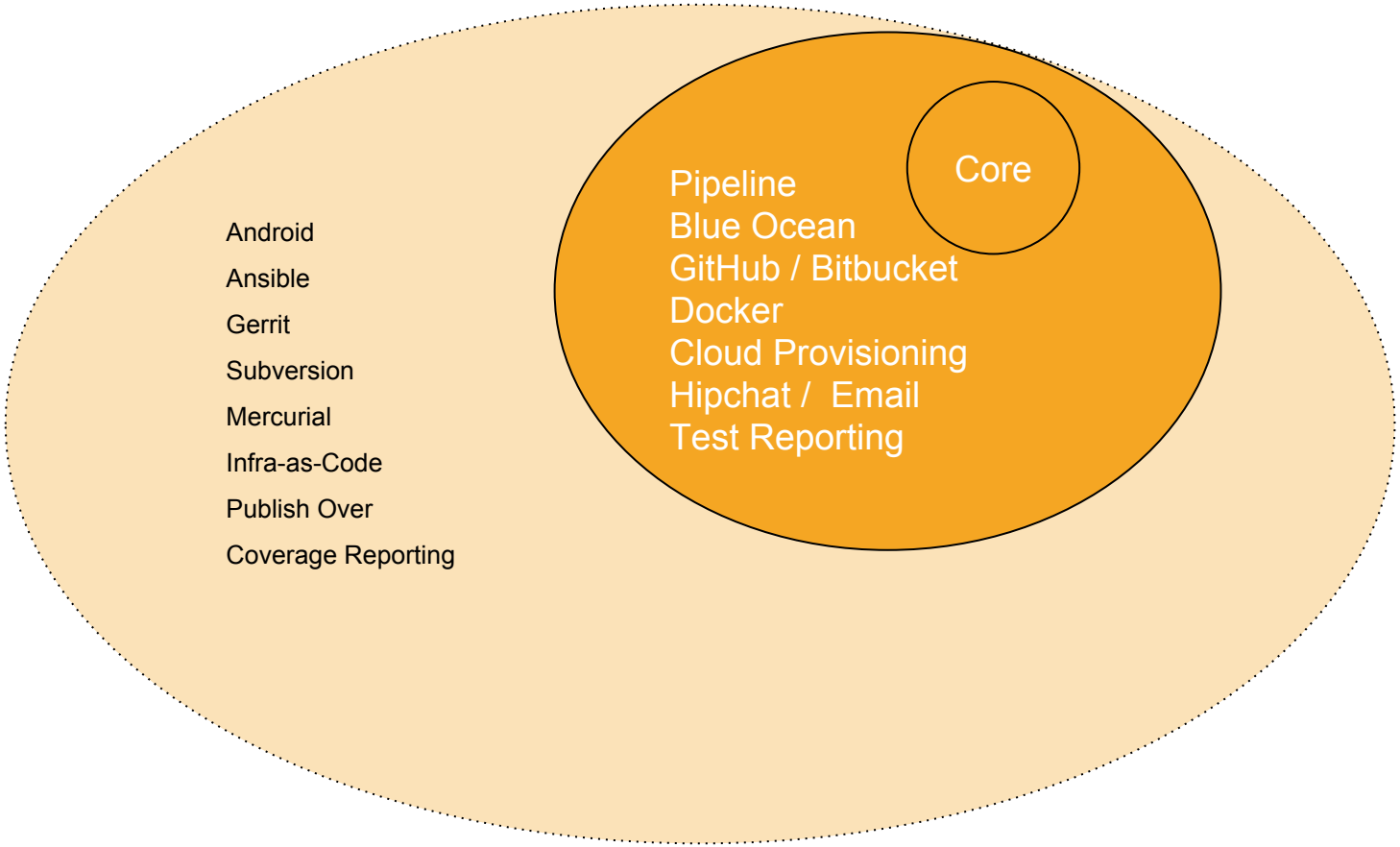


A user should understand the unique value of Jenkins in: 5 minutes and 5 clicks

The Goal

- For new, and existing, Jenkins users:
  - "Jenkins with the batteries included."
  - Help a user go from "zero" to productive, and understanding the unique value of Jenkins in under five clicks and five minutes.
- For the Jenkins project:
  - Platform for viewing Jenkins as "the whole is greater than the sum of its parts."

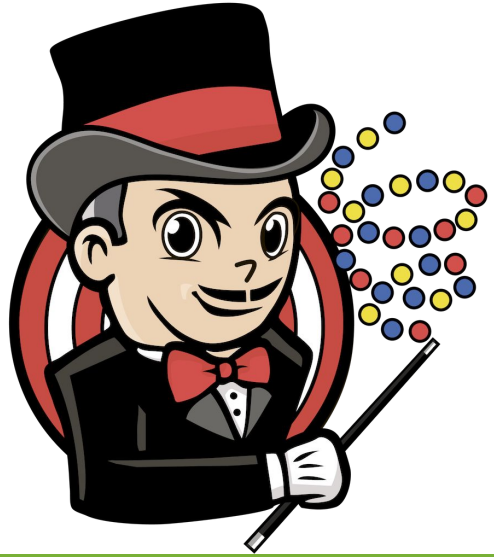




"Brighter Line" of Features

Project Evergreen

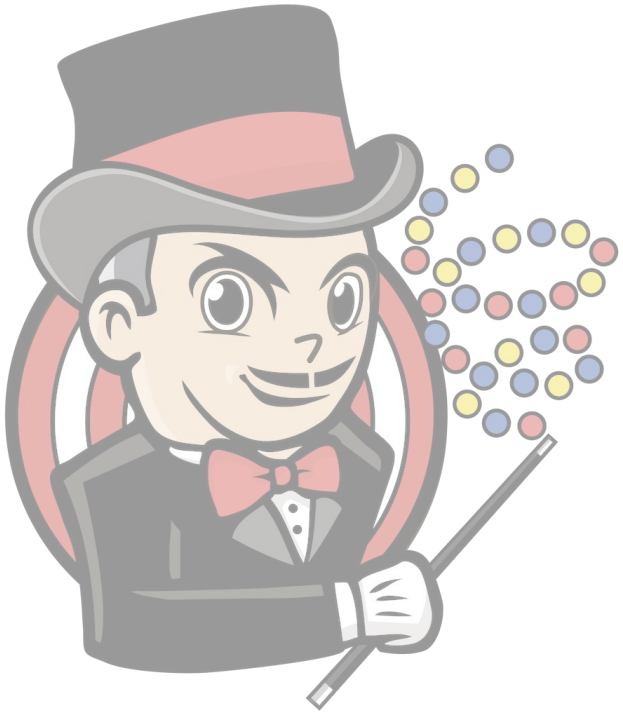
How we get there



Audience

Those Who Get Shit Done

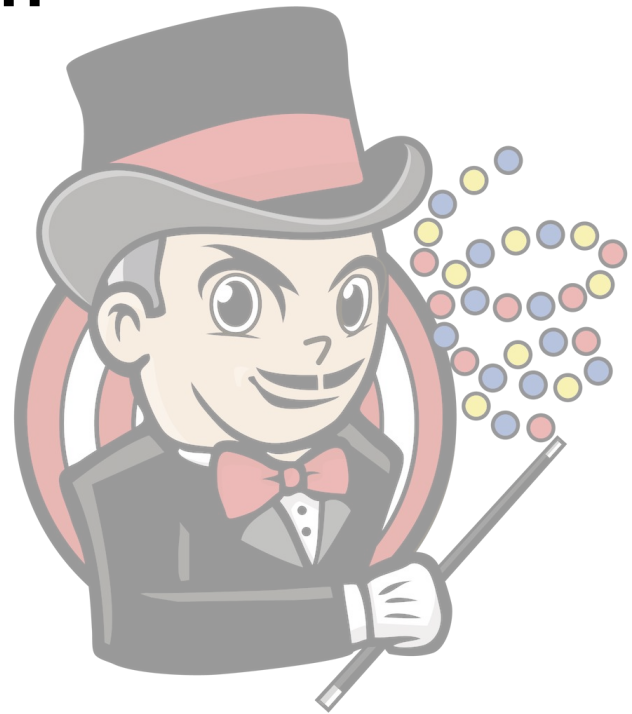
- Jenkins is for developers who want to Get Shit Done.
  - Don't want to adopt a maintenance burden just to have CI/CD.
  - Don't have time/energy/desire to become "experts" in setting up Jenkins.
  - Status quo requires *significant* effort to provision and configure a "good" Jenkins environment.
- We're competing against services which enable (roughly) one-click-to-CI
  - Continuous Evolution: tools should transparently get better over time.

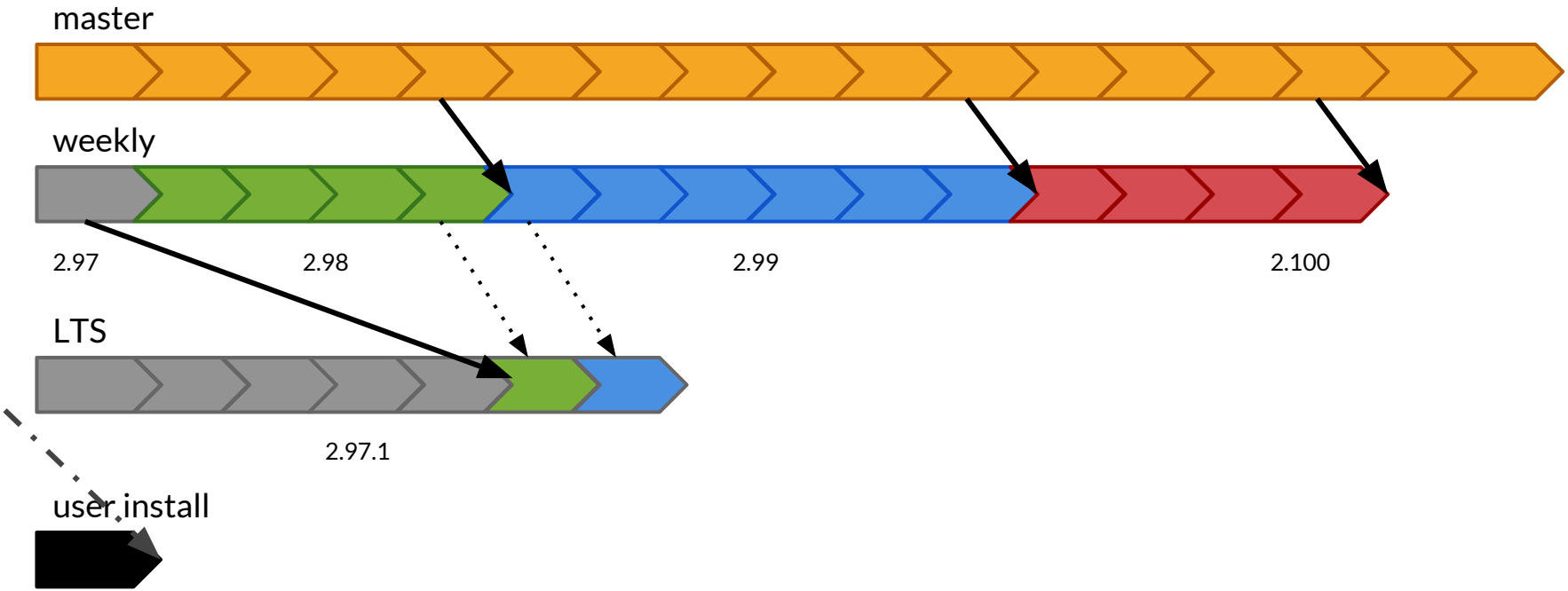


What is Project Evergreen

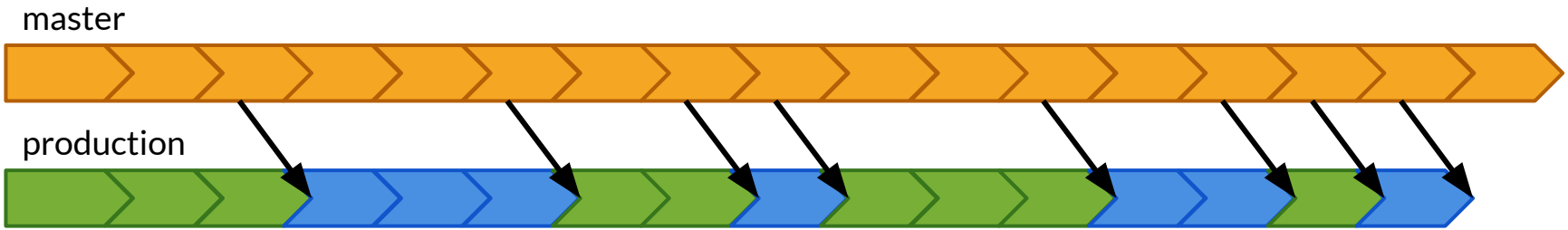
Rolling, auto-updating, distribution

- "Chrome-ification of Jenkins"
  - Applying continuous delivery, rather than legacy train, model.
- Batteries included
  - Core, and key plugins all part and parcel in the distribution.
    - Pipeline, Blue Ocean, etc.
  - Dramatically reduced version matrices and test workload.
- Live error reporting and instance telemetry
  - Instantaneous, automatic, feedback for new deployments.
- Feature-flagging and dark-launching
  - Enable experimental code locally or on cohort.

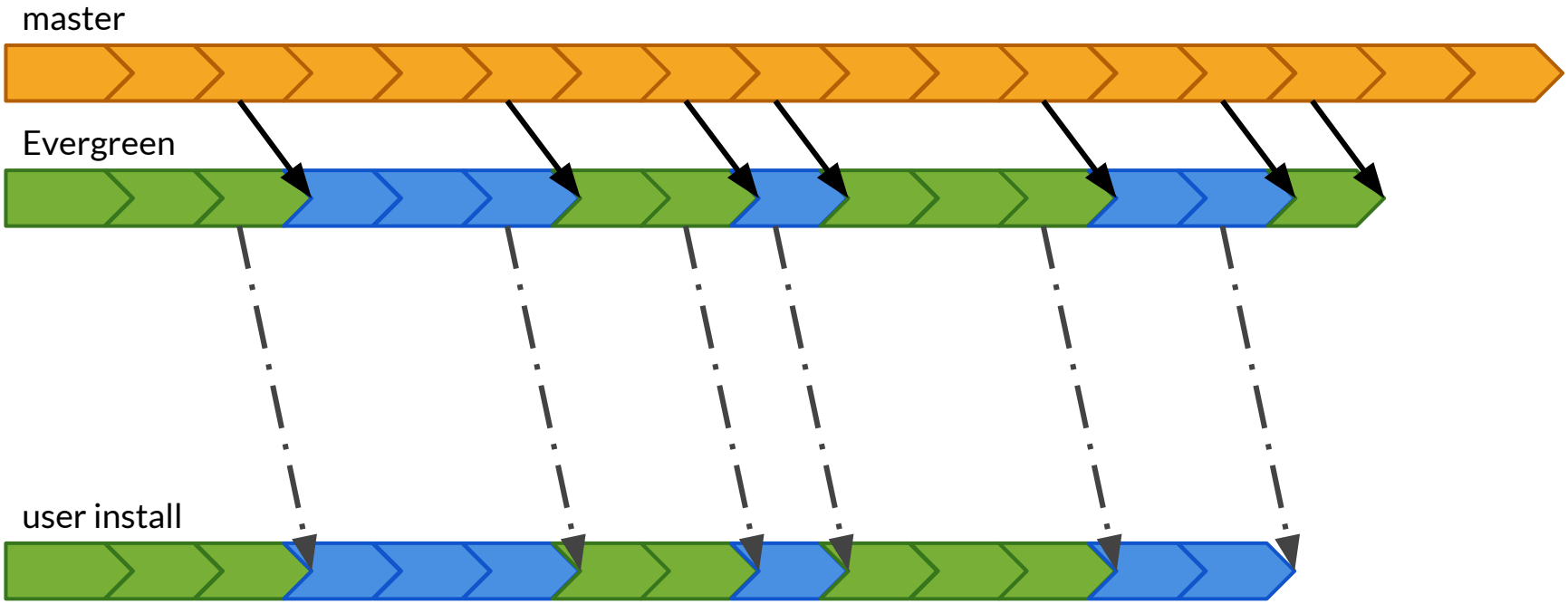




Slow, train delivery for Jenkins



Continuous delivery for web applications

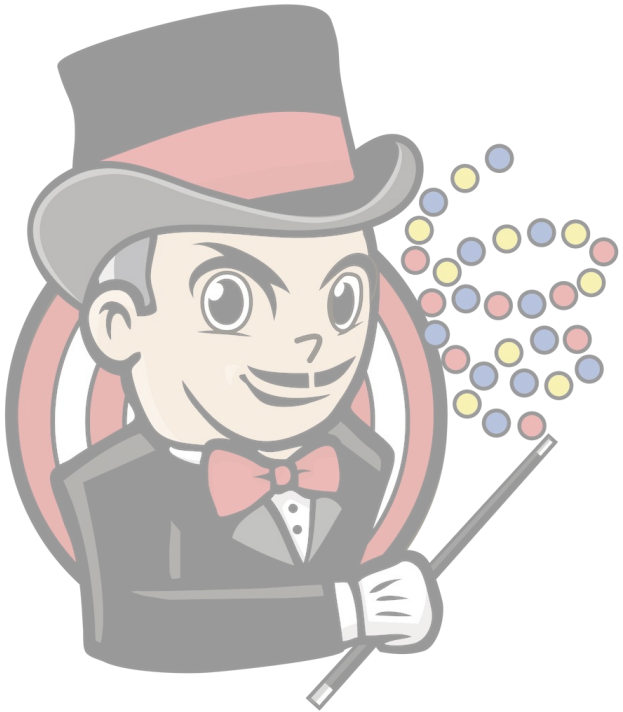


Continuous delivery for Jenkins

Plus Jenkins Essentials

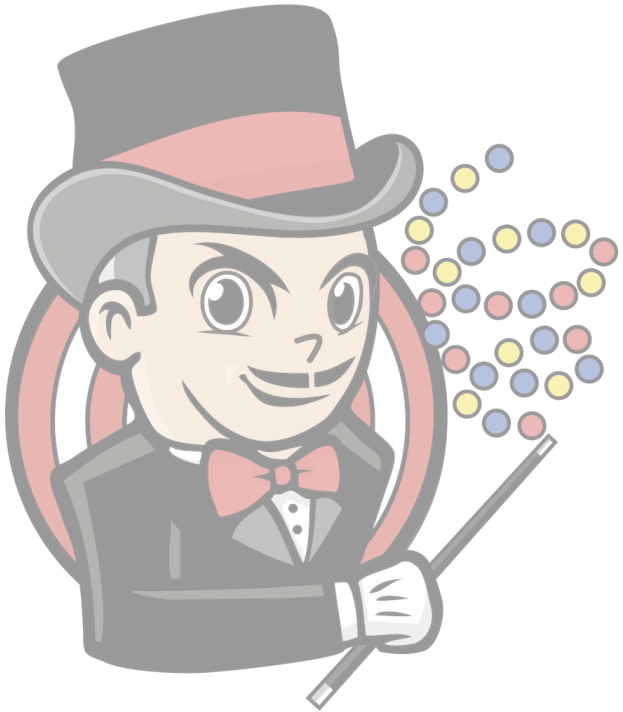
Automatic Sane Defaults

- Defaults for Evergreen automated to reduce as many opportunities for manual user configuration.
- Reduce configuration redundancies.
  - Users shouldn't have to redundantly enter any configuration.
- Intelligently identify environmental capabilities.
  - In a cloud environment, enable cloud-specific defaults.
  - In a container environment, enable container-specific defaults.



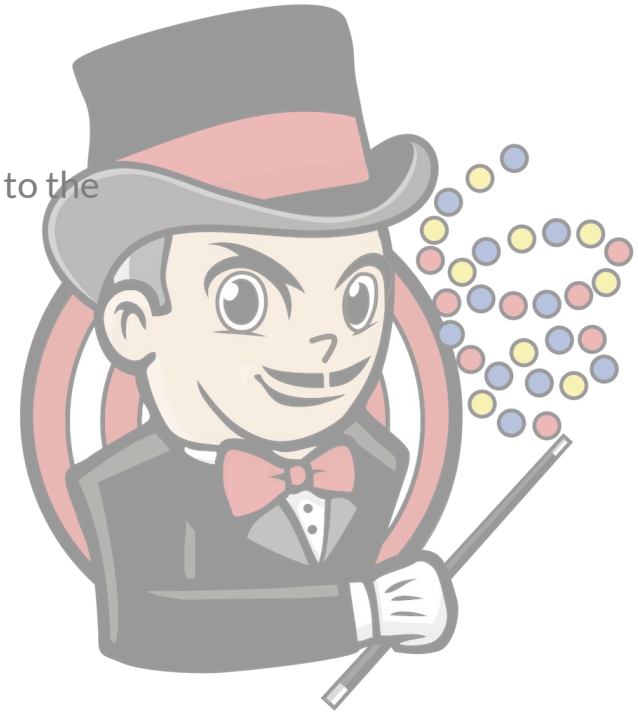
Connected

- Connected to Jenkins services for more seamless functionality
    - Transparently connect Jenkins-to-services such as GitHub webhooks
- Authentication/Authorization brokerage
    - "Sign in with GitHub"
    - "Sign in with Google"
- Error telemetry/user analytics telemetry.
- Make Jenkins feel as easy and connected as a SaaS offering.

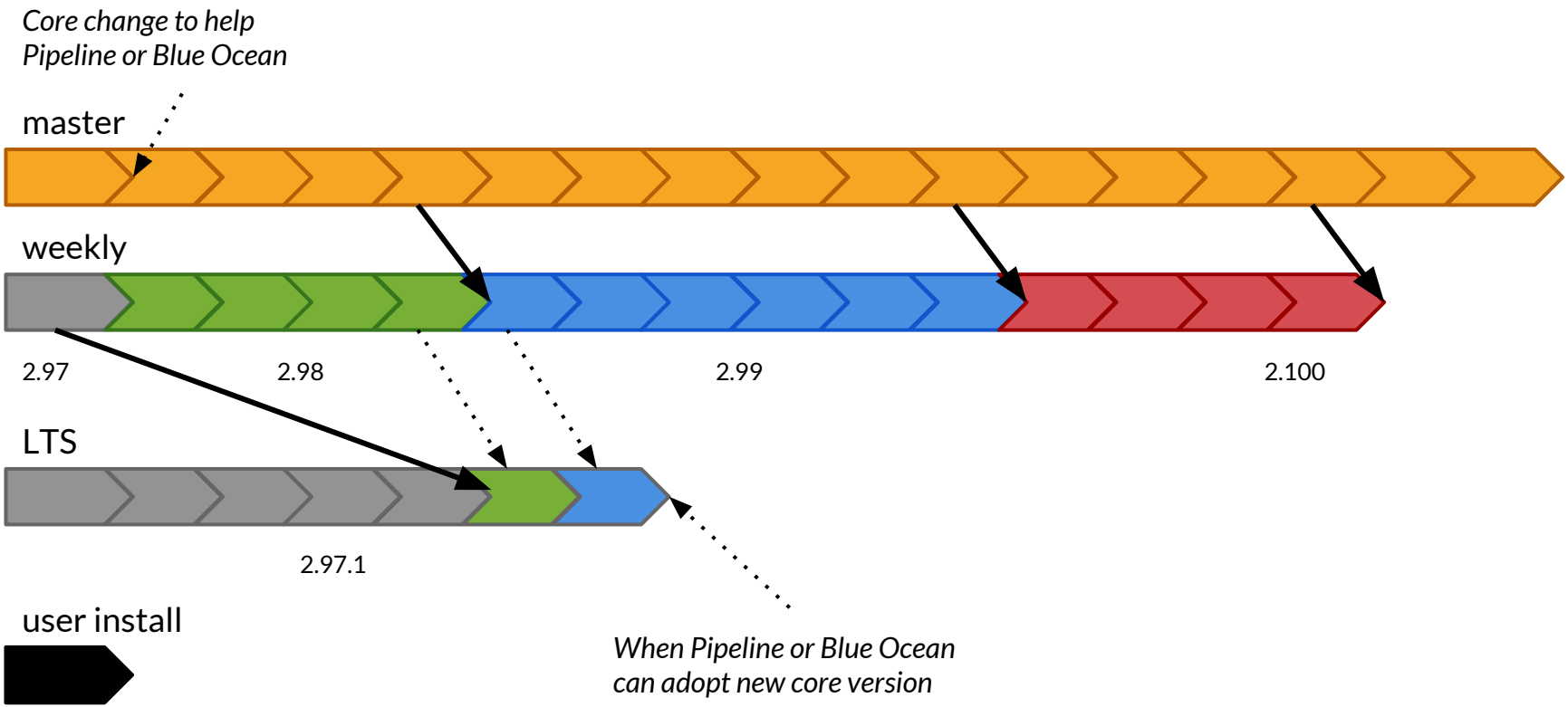


Obvious path to success

- Baked-in documentation and example Pipelines
    - Successful use of Jenkins should be eminently understandable to the end-user
- Disable/hide legacy Jenkins functionality.
- User-visible errors should be actionable and understandable.



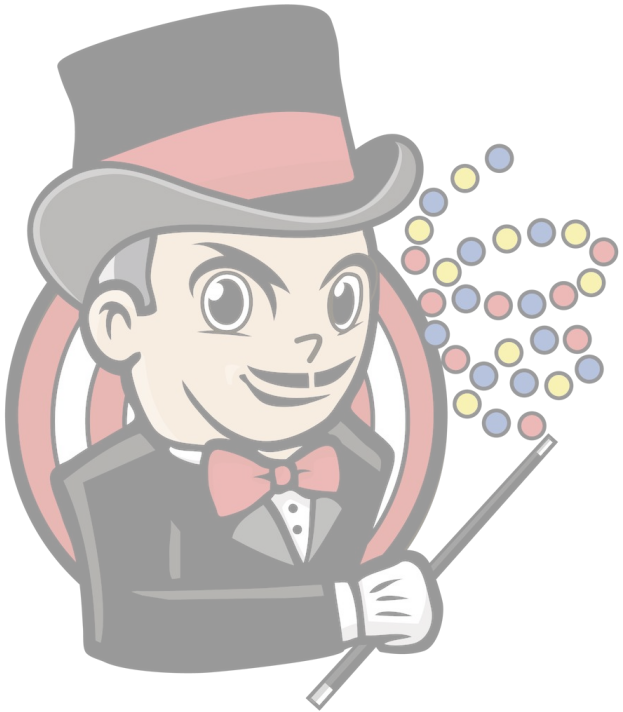
Why we should build it



Best case: ~4 weeks
Worst case: ~3 months

Developer Velocity

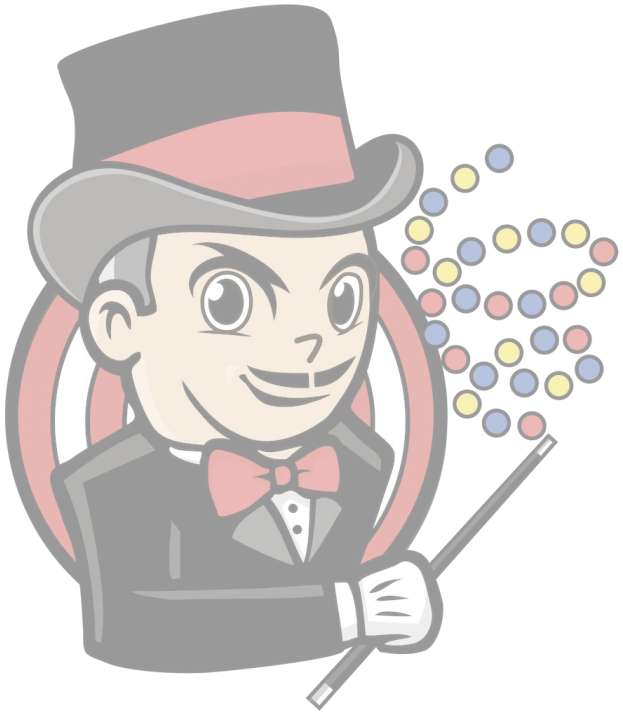
- Changes to Jenkins core don't receive significant feedback until ~1 week after merge to master.
  - E.g. JEP-200 related changes.
- Changes to Jenkins core, which plugins anticipate, cannot be adopted for months.



Challenges and Risks

Challenges

- Culture shift in the project
  - No guarantee other participants, notably Essentials plugin developers, are interested in it.
-



Thanks!