

How NOT to Analyze your Release Process

... with Suggestions!



Bram Adams
bram.adams@polymtl.ca

Modern Release Engineering in a Nutshell

Why Researchers should Care



Bram Adams
Polytechnique Montréal, Canada
bram.adams@polymtl.ca



Shane McIntosh
McGill University, Canada
shane.mcintosh@mcgill.ca

Abstract—The release engineering process is the process that brings high quality code changes from a developer’s workspace to the end user, encompassing code change integration, continuous integration, build system specifications, infrastructure-as-code, deployment and release. Recent practices of continuous delivery, which bring new content to the end user in days or hours rather than months or years, have generated a surge of industry-driven interest in the release engineering pipeline. This paper argues that the involvement of researchers is essential, by providing a brief introduction to the six major phases of the release engineering pipeline, a roadmap of future research, and a checklist of three major ways that the release engineering process of a system under study can invalidate the findings of software engineering studies. The main take-home message is that, while release engineering technology has flourished tremendously due to industry, empirical validation of best practices and the impact of the release engineering process on (amongst others) software quality is largely missing and provides major research opportunities.

I. INTRODUCTION

Release engineering is the process responsible for taking the individual code contributions of developers and bringing those to the end user in the form of a high quality software release. From start to finish, a myriad of tasks need to be performed by

Mozilla Firefox [71] and the Facebook Mobile app have a release “cycle time” of 2-6 weeks, while web-delivered content like the Netflix and Facebook websites push new releases 1-2 times daily [65]. Furthermore, lean web apps like the popular IMVU chat application¹ release up to 50 times per day [26].

As these pioneering organizations successfully developed experimental tools and practices to make such rapid release cycles a reality, Facebook’s release engineering director, Chuck Rossi, claimed that “continuous delivery for web apps is a solved problem” [65]. However, he did add “..., yet continuous delivery for mobile apps is a serious challenge.” Indeed, for every success and breakthrough that has been made, there are a slew of failures. Even today, software organizations who are not at the forefront of the release engineering revolution need to consider what release practices should be adopted, what tools they should invest in and what profiles should be used to make hiring decisions. Even for the pioneers of modern release engineering, newer technologies like mobile apps still pose open challenges. Broader questions include: what will be the long-term effect on user-perceived quality of releases [43, 45], how quickly will technical debt ramp up

Modern Release Engineering in a Nutshell

Why Researchers should Care



Bram Adams
Polytechnique Montréal, Canada
bram.adams@polymtl.ca



Shane McIntosh
McGill University, Canada
shane.mcintosh@mcgill.ca

@ FoSE,
SANER 2016

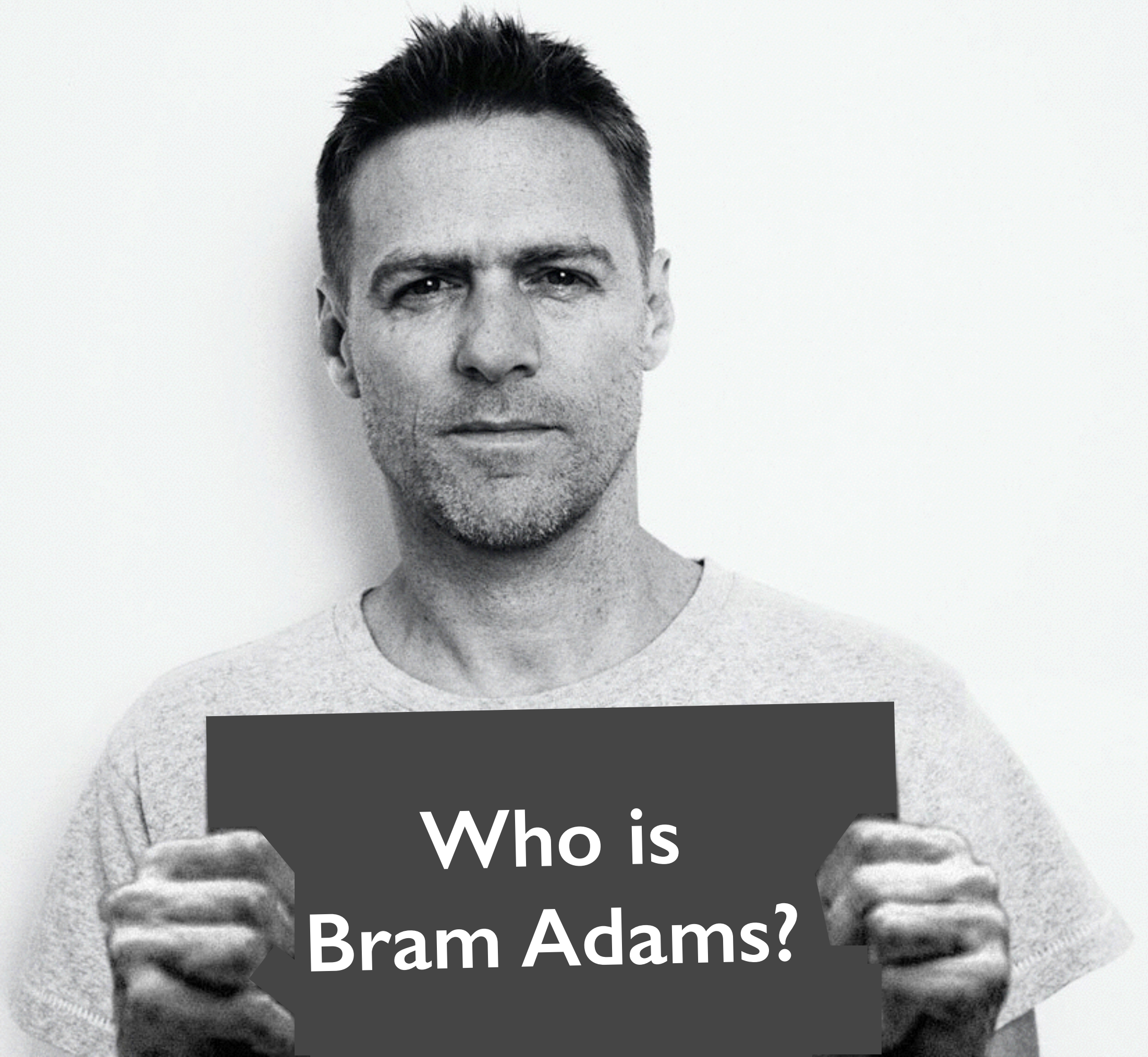
Abstract—The release engineering process is the one that brings high quality code changes from a developer’s machine to the end user, encompassing code change integration, continuous integration, build system specifications, infrastructure-as-code, deployment and release. Recent practices of continuous delivery, which bring new content to the end user in days or hours rather than months or years, have generated a surge of industry-driven interest in the release engineering pipeline. This paper argues that the involvement of researchers is essential, by providing a brief introduction to the six major phases of the release engineering pipeline, a roadmap of future research, and a checklist of three major ways that the release engineering process of a system under study can invalidate the findings of software engineering studies. The main take-home message is that, while release engineering technology has flourished tremendously due to industry, empirical validation of best practices and the impact of the release engineering process on (amongst others) software quality is largely missing and provides major research opportunities.

I. INTRODUCTION

Release engineering is the process responsible for taking the individual code contributions of developers and bringing those to the end user in the form of a high quality software release. From start to finish, a myriad of tasks need to be performed by

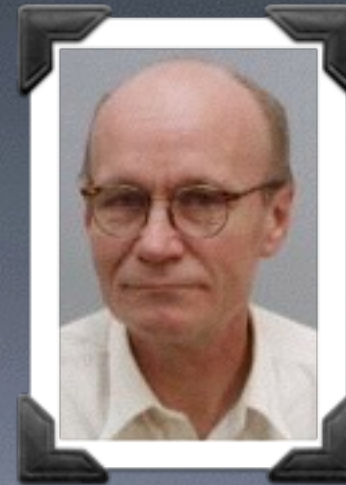
and the Facebook Mobile app have a release cycle of 2-6 weeks, while web-delivered content like the Netflix and Facebook websites push new releases 1-2 times daily [65]. Furthermore, lean web apps like the popular IMVU chat application¹ release up to 50 times per day [26].

As these pioneering organizations successfully developed experimental tools and practices to make such rapid release cycles a reality, Facebook’s release engineering director, Chuck Rossi, claimed that “continuous delivery for web apps is a solved problem” [65]. However, he did add “..., yet continuous delivery for mobile apps is a serious challenge.” Indeed, for every success and breakthrough that has been made, there are a slew of failures. Even today, software organizations who are not at the forefront of the release engineering revolution need to consider what release practices should be adopted, what tools they should invest in and what profiles should be used to make hiring decisions. Even for the pioneers of modern release engineering, newer technologies like mobile apps still pose open challenges. Broader questions include: what will be the long-term effect on user-perceived quality of releases [43, 45], how quickly will technical debt ramp up



**Who is
Bram Adams?**





Herman Tromp
Ghent University



Wolfgang De Meuter
Vrije Universiteit Brussel



Ahmed E. Hassan
Queen's University



M
C·I·S
I·S

(Lab on Maintenance,
Construction and Intelligence
of Software)





M
C·I·S

Mohammed

Parisa

Alexandre

Bram

Parastou

Mahdis

Jojo

M
C·I·S

<http://mcis.polymtl.ca>

Back in 2009 ...



Express yourself in the world's largest 3D Chat and Dress-Up community!

[Member Login](#)



113,653 people online now

in **88** different countries!

Sign in with:



Choose Your FREE Avatar



Over 2 Million people
like IMVU on Facebook!

FREE



Express yourself in the world's largest 3D Chat and Dress-Up community!

Member Login

On average, we
release new code **fifty**
times a day.

 **113,653** people online now
in **88** different countries!

Sign in with:



Choose Your **FREE** Avatar 

Over 2 Million people
like IMVU on Facebook!

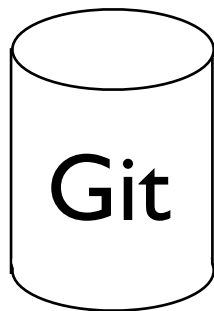
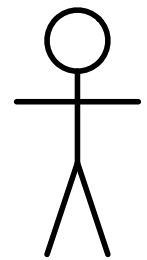
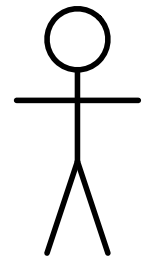
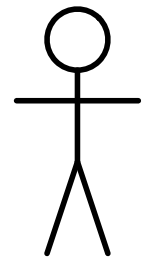
FREE



How to release 50 times/day?

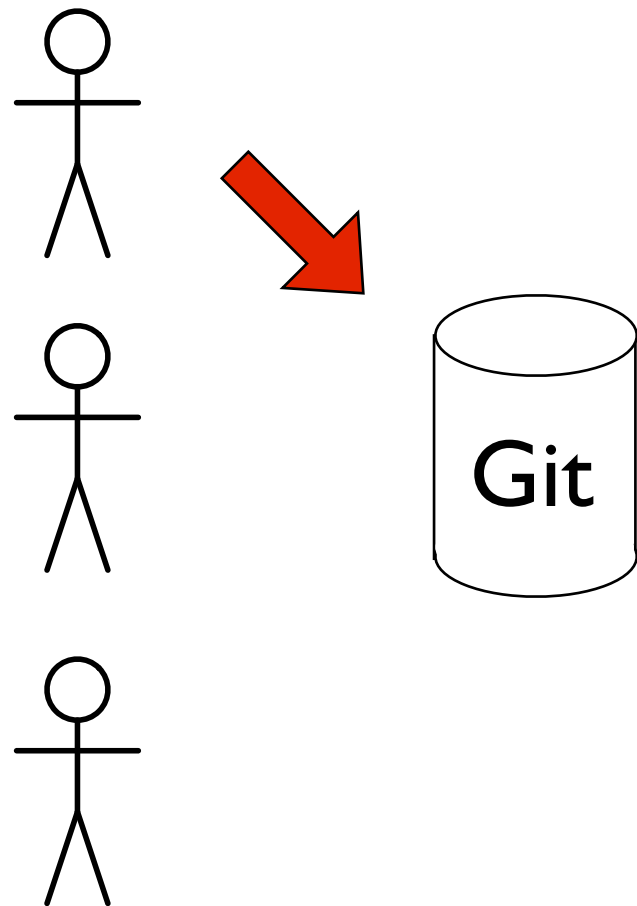


How to release 50 times/day?





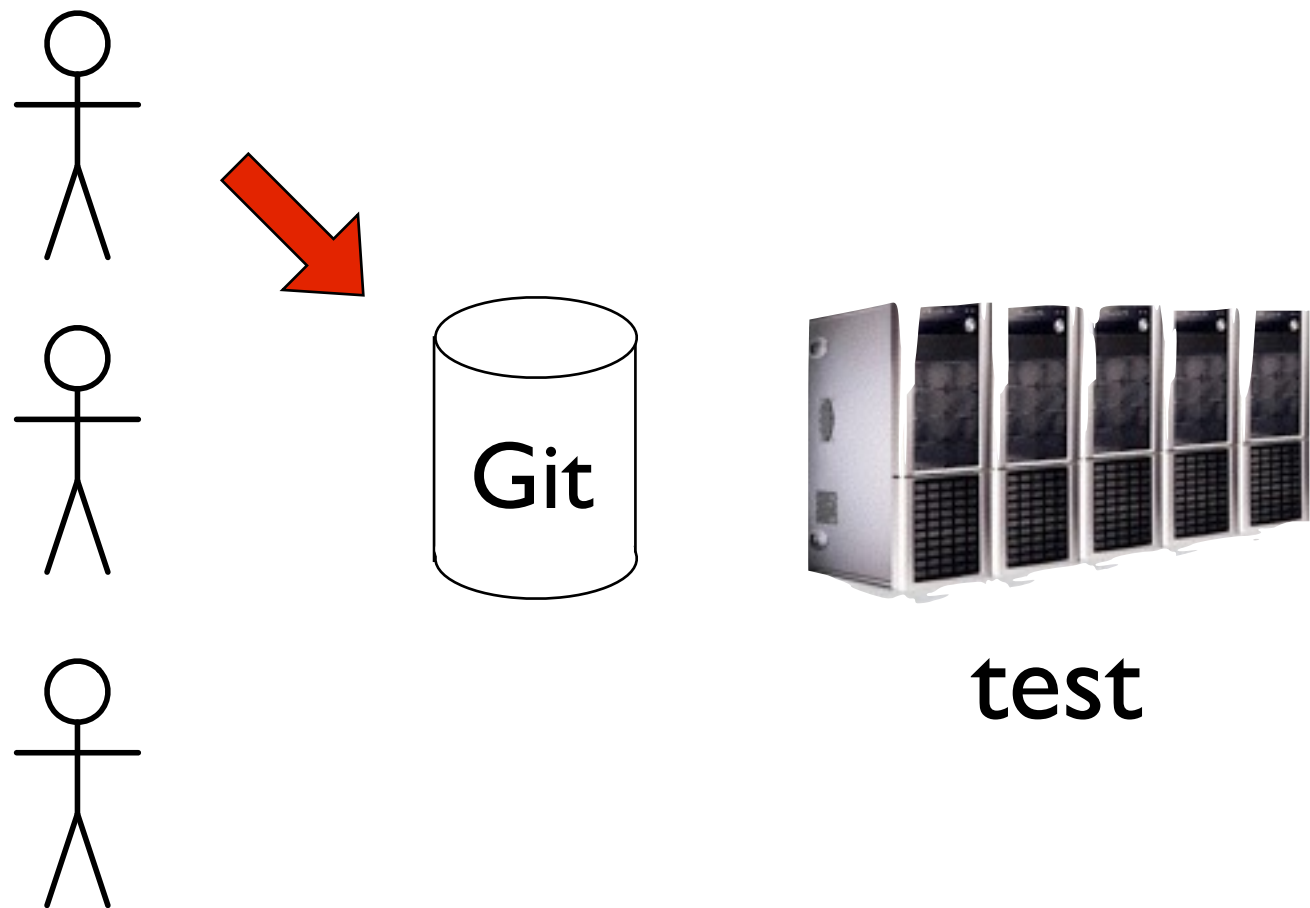
How to release 50 times/day?



continuous
integration



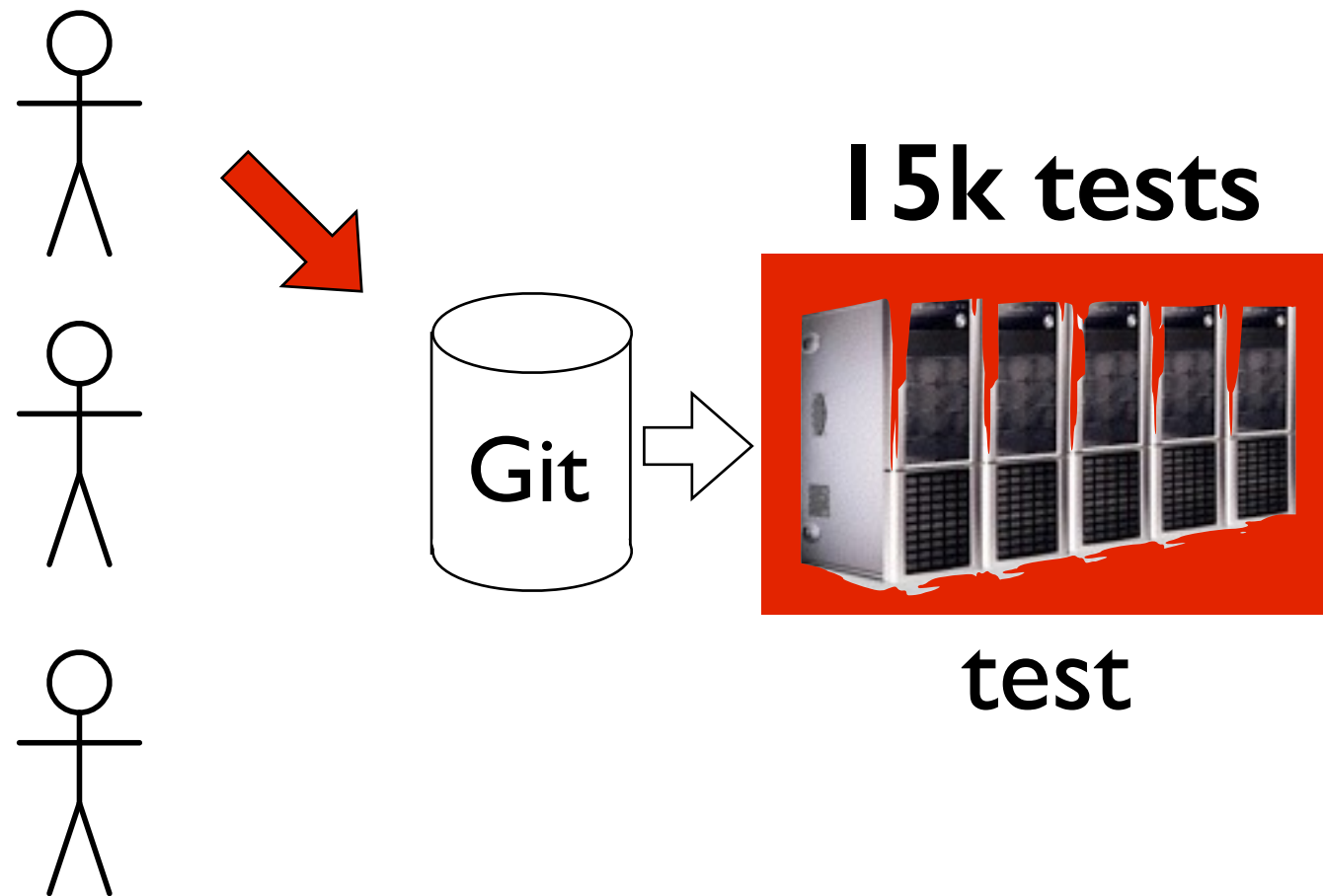
How to release 50 times/day?



continuous
integration



How to release 50 times/day?

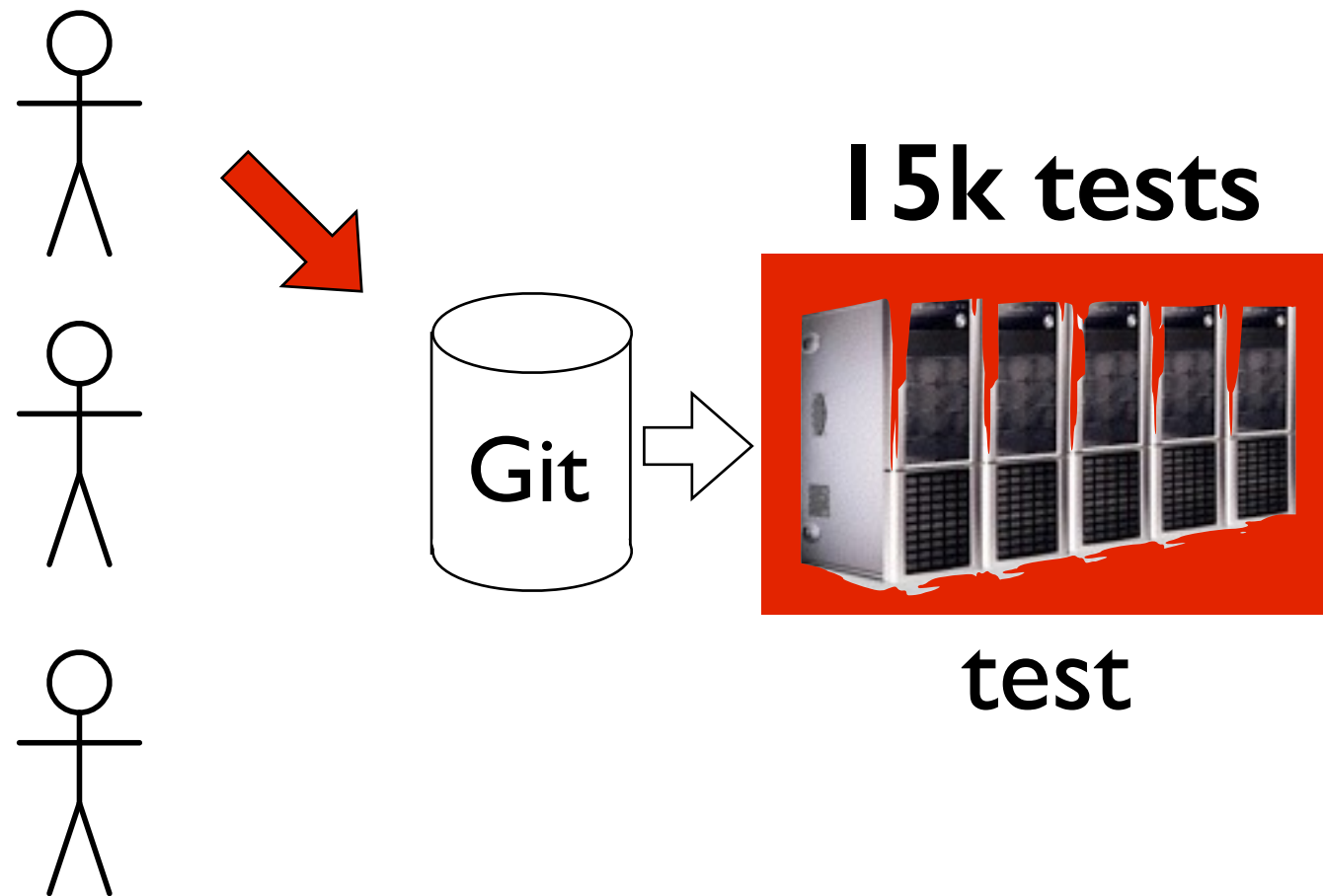


continuous
integration

9 min.



How to release 50 times/day?



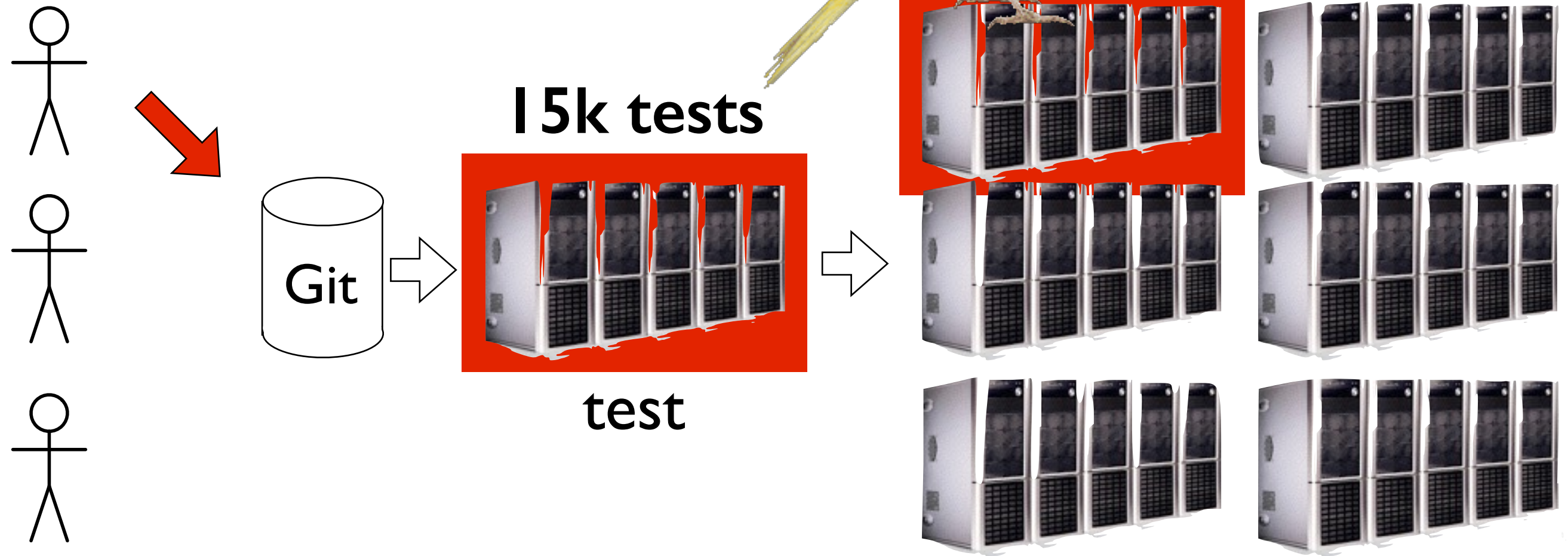
staging/production

continuous
integration

9 min.



How to release 50 times/day?

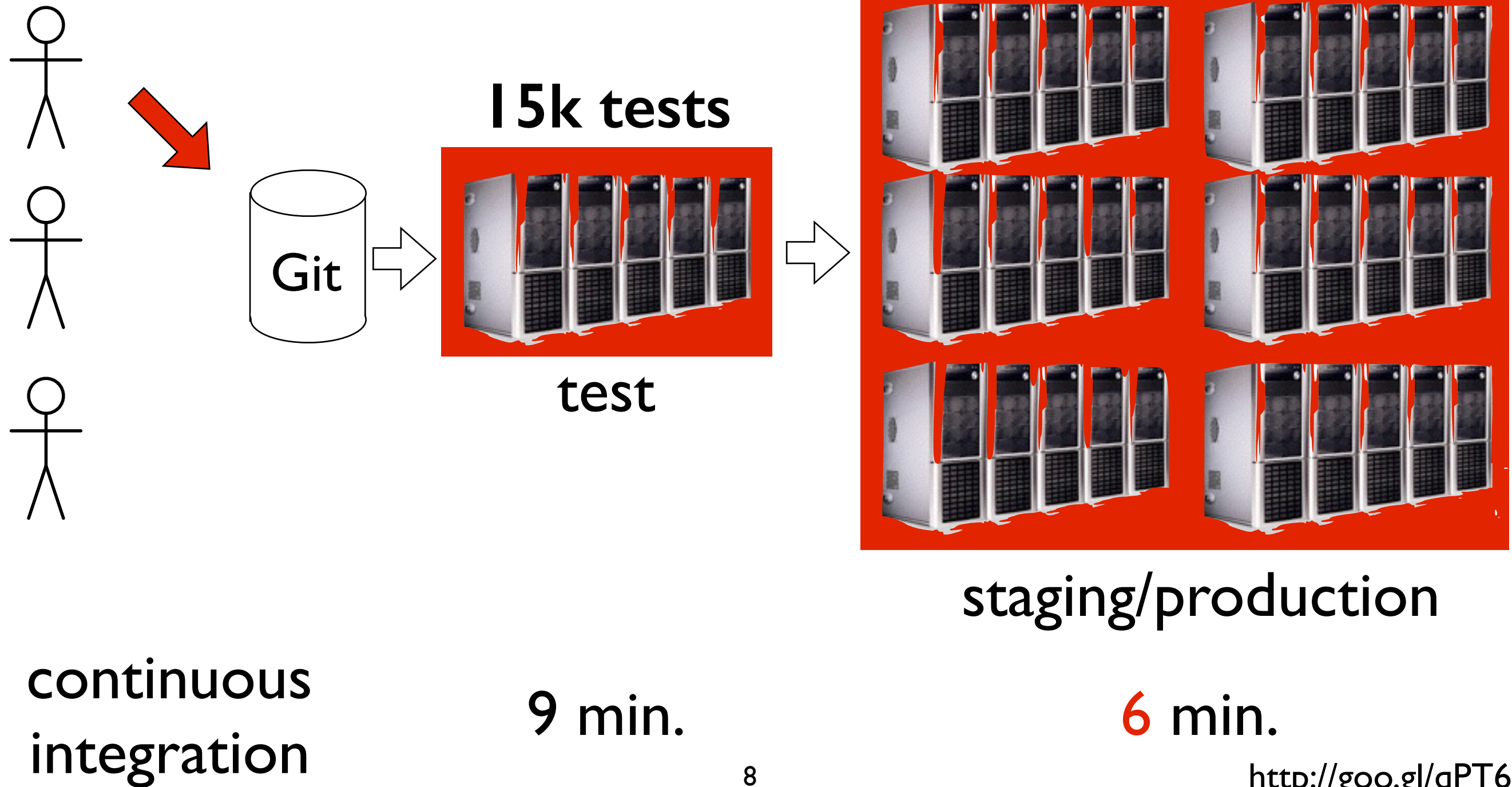


continuous
integration

9 min.



How to release 50 times/day?

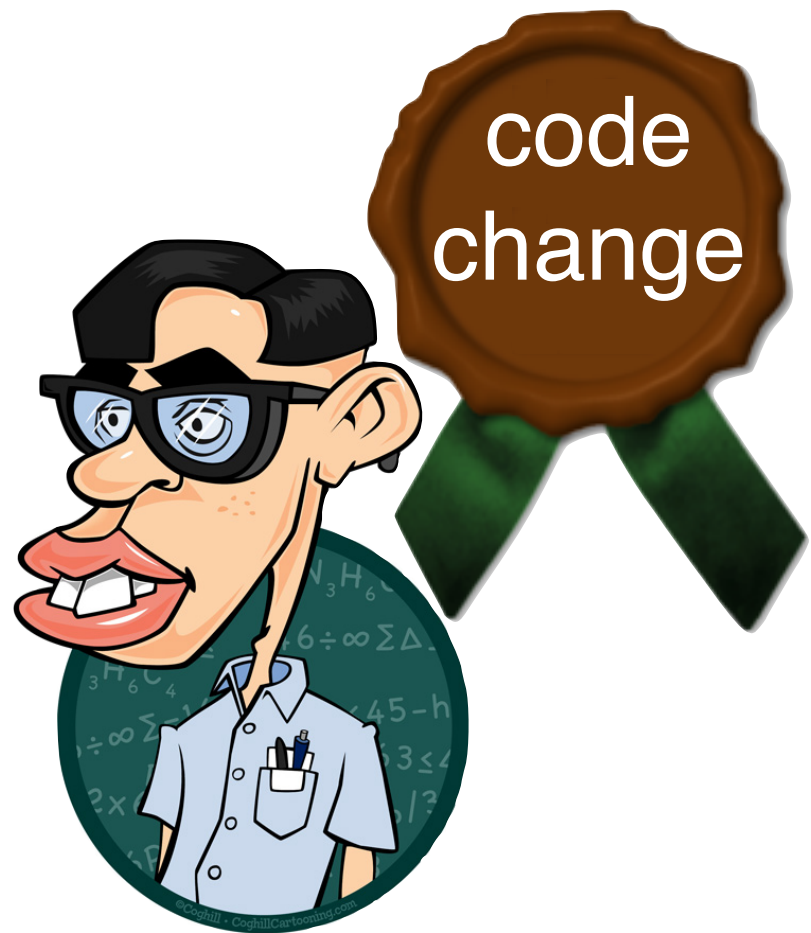


Release engineering aims to ...

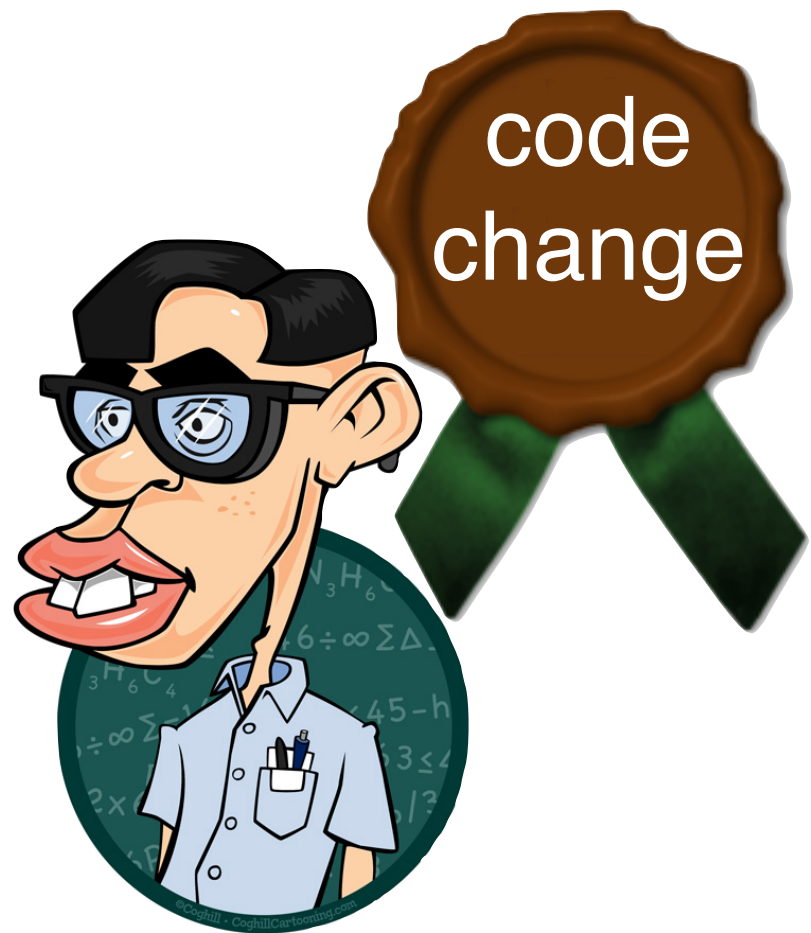
Release engineering aims to ...



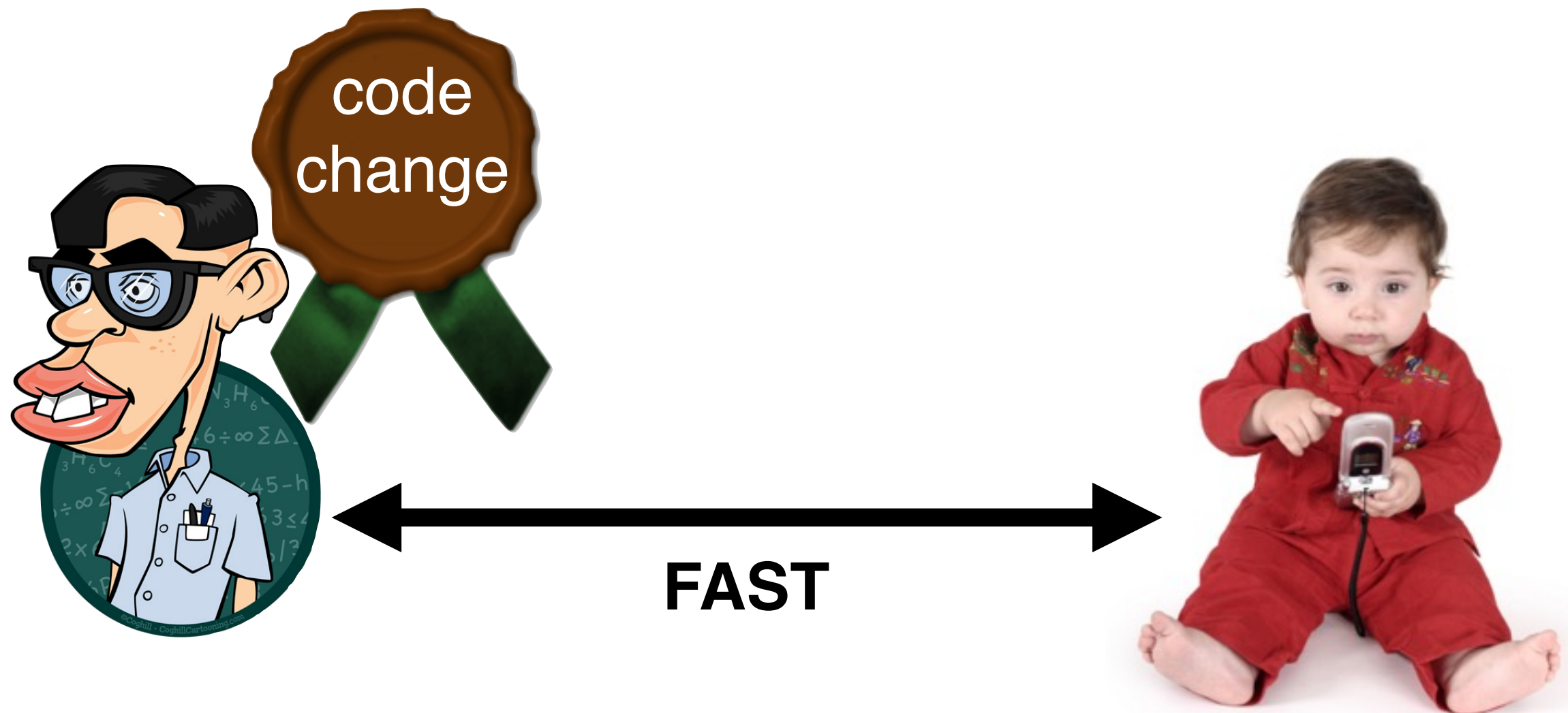
Release engineering aims to ...



Release engineering aims to ...

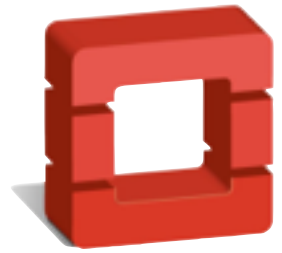


Release engineering aims to ...



Nowadays ...

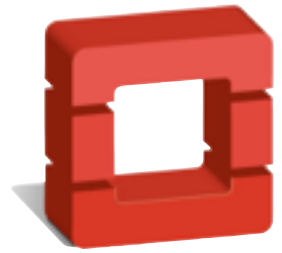
Time-boxed releases



openstack™

6 months

Time-boxed releases



openstack™

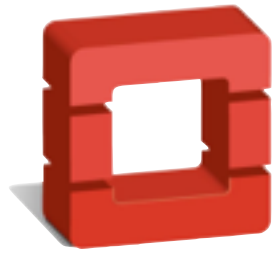
6 months



6 weeks

Time-boxed releases

Time-boxed releases



openstack™

6 months

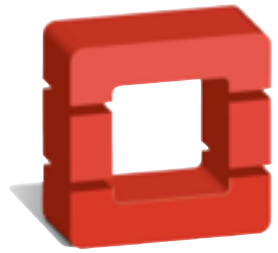


6 weeks



2 weeks
(mobile)

Time-boxed releases



openstack™

6 months



6 weeks

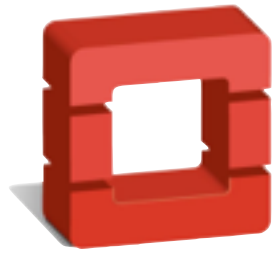


2 weeks
(mobile)



twice/day
(web)

Time-boxed releases



openstack™

6 months



6 weeks



2 weeks
(mobile)



twice/day
(web)

NETFLIX

daily

TIMELINE

6 weeks

Awesomeness lands
on Firefox Nightly

6 weeks

Stabilize on
Firefox Aurora

More awesomeness
on Firefox Nightly

6 weeks

Stabilize on
Firefox Beta

Stabilize on
Firefox Aurora

Even more
awesomeness
on Firefox Nightly

6 weeks

Firefox
Release!

Stabilize on
Firefox Beta

Stabilize on
Firefox Aurora

6 weeks

Firefox
Release!

Stabilize on
Firefox Beta

Firefox
Release!

Release Trains

TIMELINE

6 weeks

Awesomeness lands
on Firefox Nightly

6 weeks

Stabilize on
Firefox Aurora

More awesomeness
on Firefox Nightly

6 weeks

Stabilize on
Firefox Beta

Stabilize on
Firefox Aurora

Even more
awesomeness
on Firefox Nightly

6 weeks

Firefox
Release!

Stabilize on
Firefox Beta

Stabilize on
Firefox Aurora

6 weeks

Firefox
Release!

Stabilize on
Firefox Beta

Firefox
Release!

Release Trains

TIMELINE

6 weeks

Awesomeness lands
on Firefox Nightly

6 weeks

Stabilize on
Firefox Aurora

More awesomeness
on Firefox Nightly

6 weeks

Stabilize on
Firefox Beta

Stabilize on
Firefox Aurora

Even more
awesomeness
on Firefox Nightly

6 weeks

Firefox
Release!

Stabilize on
Firefox Beta

Stabilize on
Firefox Aurora

6 weeks

Firefox
Release!

Stabilize on
Firefox Beta

Firefox
Release!

Release Trains

TIMELINE

6 weeks

Awesomeness lands
on Firefox Nightly

6 weeks

Stabilize on
Firefox Aurora

More awesomeness
on Firefox Nightly

6 weeks

Stabilize on
Firefox Beta

Stabilize on
Firefox Aurora

Even more
awesomeness
on Firefox Nightly

6 weeks

**Firefox
Release!**

Stabilize on
Firefox Beta

Stabilize on
Firefox Aurora

6 weeks

**Firefox
Release!**

Stabilize on
Firefox Beta

**Firefox
Release!**

Release Trains

TIMELINE

6 weeks

Awesomeness lands
on Firefox Nightly

6 weeks

Stabilize on
Firefox Aurora

6 weeks

Stabilize on
Firefox Beta

6 weeks

**Firefox
Release!**

6 weeks

*More awesomeness
on Firefox Nightly*

Stabilize on
Firefox Aurora

Stabilize on
Firefox Beta

**Firefox
Release!**

*Even more
awesomeness
on Firefox Nightly*

Stabilize on
Firefox Aurora

Stabilize on
Firefox Beta

**Firefox
Release!**

Release Trains

TIMELINE

6 weeks

Awesomeness lands
on Firefox Nightly

6 weeks

Stabilize on
Firefox Aurora

More awesomeness
on Firefox Nightly

6 weeks

Stabilize on
Firefox Beta

Stabilize on
Firefox Aurora

*Even more
awesomeness
on Firefox Nightly*

6 weeks

**Firefox
Release!**

Stabilize on
Firefox Beta

Stabilize on
Firefox Aurora

6 weeks

**Firefox
Release!**

Stabilize on
Firefox Beta

**Firefox
Release!**

Release Trains

TIMELINE

6 weeks

Awesomeness lands
on Firefox Nightly

6 weeks

Stabilize on
Firefox Aurora

More awesomeness
on Firefox Nightly

6 weeks

Stabilize on
Firefox Beta

Stabilize on
Firefox Aurora

Even more
awesomeness
on Firefox Nightly

6 weeks

Firefox
Release!

Stabilize on
Firefox Beta

Stabilize on
Firefox Aurora

6 weeks

Firefox
Release!

Stabilize on
Firefox Beta

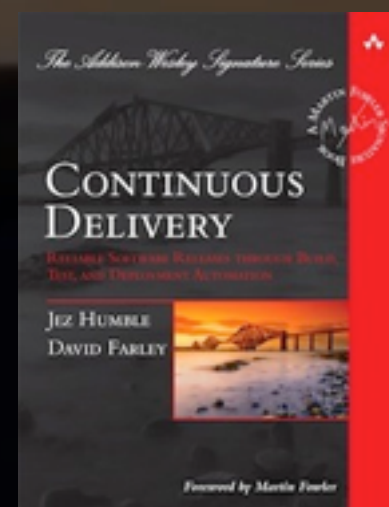
Firefox
Release!

Release Trains

Jez Humble

ThoughtWorks

<http://www.informit.com/articles/article.aspx?p=1833567>

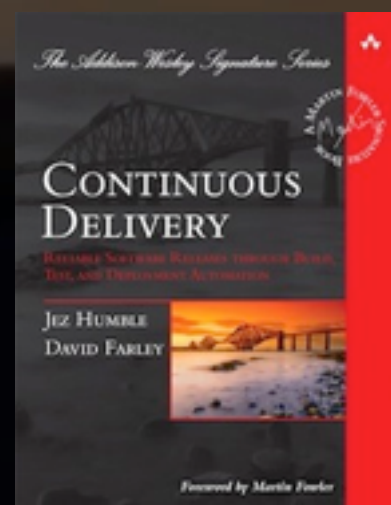



reduce the risk
of releasing software

if it hurts, do it
more frequently, and
bring the pain
forward

Jez Humble


ThoughtWorks



A profile view of Mark Zuckerberg speaking into a microphone. He has curly brown hair and is wearing a dark grey t-shirt. The background is dark and out of focus.

Mark Zuckerberg
CEO & Founder, Facebook

<http://goo.gl/UICW>

A profile view of Mark Zuckerberg speaking into a microphone. He is wearing a dark grey t-shirt. A white speech bubble is overlaid on the left side of the image.

Work fast and
don't be afraid to break
things.

Mark Zuckerberg
CEO & Founder, Facebook



Google

Build a
little and then test it.
Build some more and test
some more.

James Whittaker



Before & After

- How quickly can we ship a chemspill release?
 - ~~4-6 weeks~~ 11 hours
- How long to ship a “new feature” release?
 - ~~12-18 months~~ 6 weeks
- How many active code lines?
 - ~~1-1/2~~ 42
- How many checkins per day?
 - ~~~15 per day~~ 325 per day



Before & After

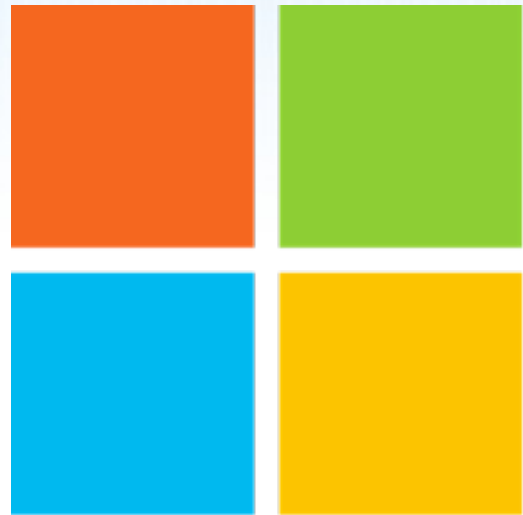
- How quickly can we ship a chemspill release?
 - ~~4-6 weeks~~ 11 hours
- How long to ship a “new feature” release?
 - ~~12-18 months~~ 6 weeks
- How many active code lines?
 - ~~1-1/2~~ 42
- How many checkins per day?
 - ~~~15 per day~~ 325 per day



Before & After

- How quickly can we ship a chemspill release?
 - ~~4-6 weeks~~ 11 hours
- How long to ship a “new feature” release?
 - ~~12-18 months~~ 6 weeks
- How many active code lines?
 - ~~1-1/2~~ 42
- How many checkins per day?
 - ~~~15 per day~~ 325 per day

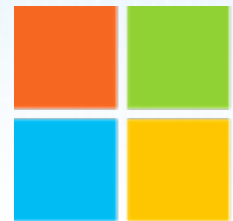
Company Size ...



Microsoft



... vs. Market Share




Microsoft



Google

XebiaLabs
Deliver Faster

BI / Monitoring 

91 En Xlr XL	92 En Ur UrbanCode Release	93 En Ls CA Service	94 En Bm BMC Release	95 En Hp HP Codar Excel	96 Pd Ex	97 En Pl Plutora Release	98 En Sr Serena Release	99 Fm Tr Trello	100 Pd Jr Jira	101 Fm Rf HipChat	102 Fm Sl Slack	103 Fm Fd Flowdock	104 Pd Pv Pivotal Tracker	105 En Sn ServiceNow
106 Os Ki Kibana	107 Fm Nr New Relic	108 Os Ni Nagios	109 Os Gg Gandlia	110 Os Ct Cacti	111 Os Gr Graphite	112 Fm Le Logentries	113 En Sp Splunk	114 Fm Sl Sumo	115 Os Ls Logstash	116 Fm Lg Loggly	117 Os Gr Graylog	118 Os Sn Snort	119 Os Tr Tripwire	120 En Cy CyberArk

OK, how should I
combine these
tools?



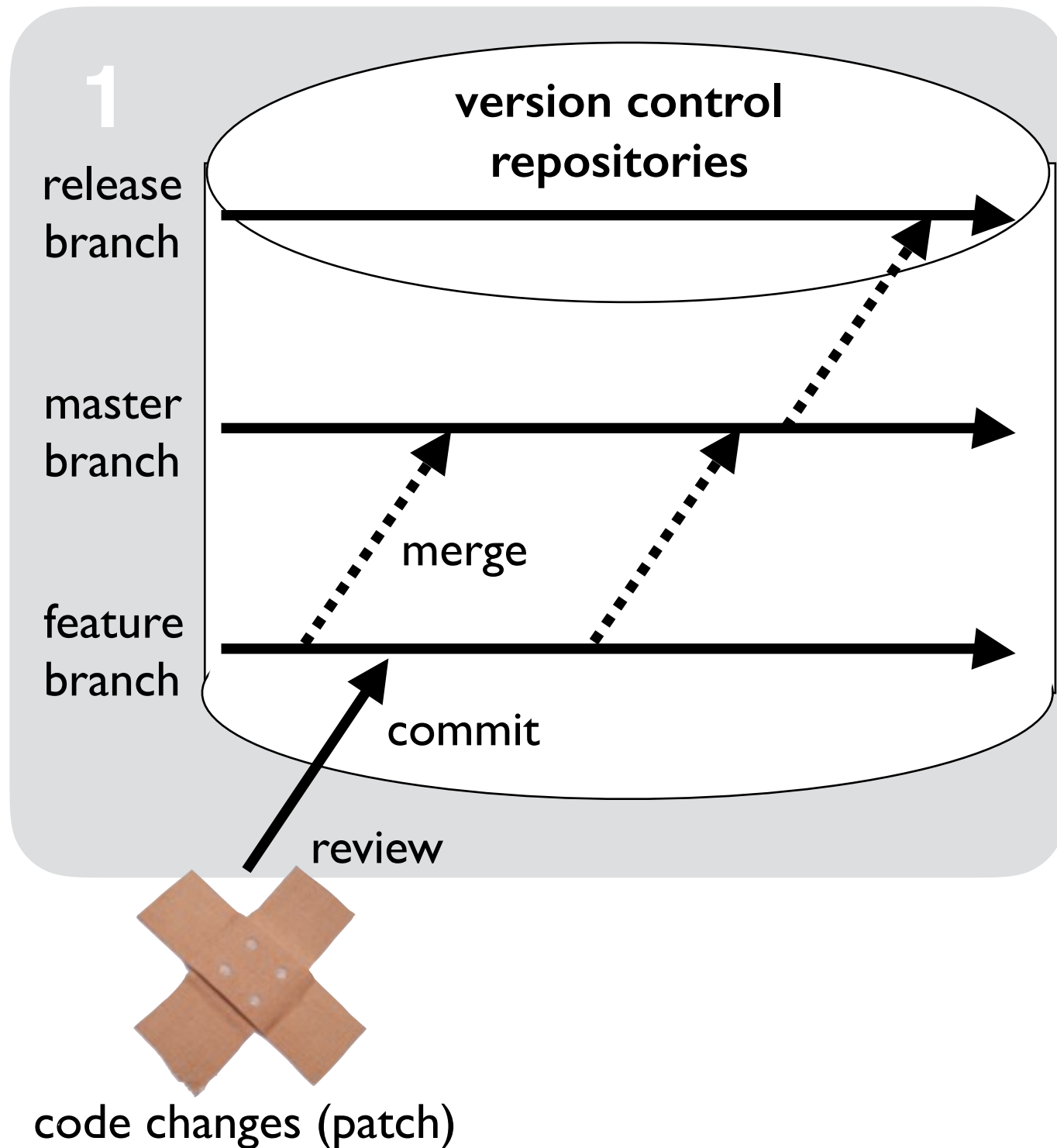
Release Engineering Pipeline (I)

Release Engineering Pipeline (I)

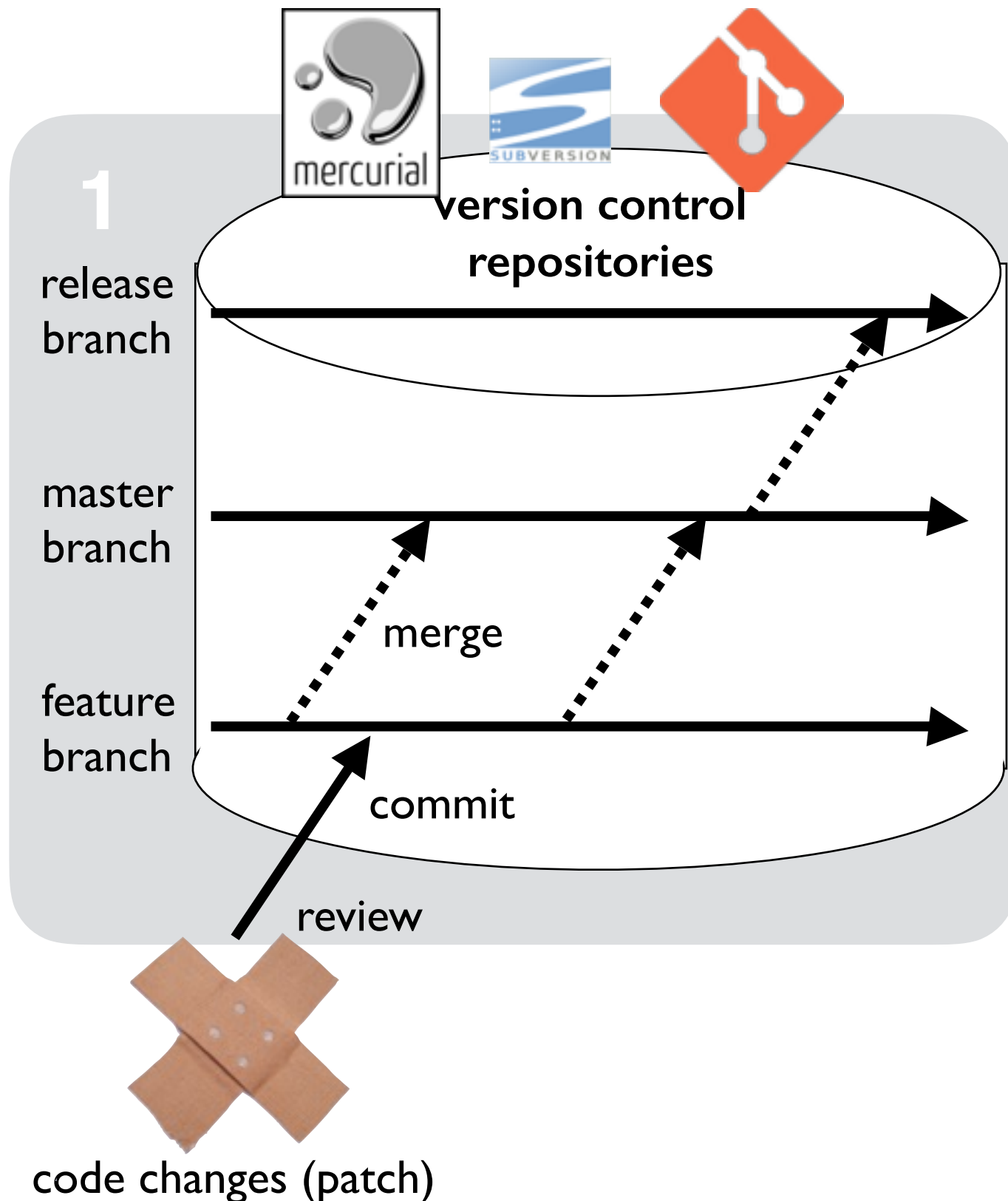


code changes (patch)

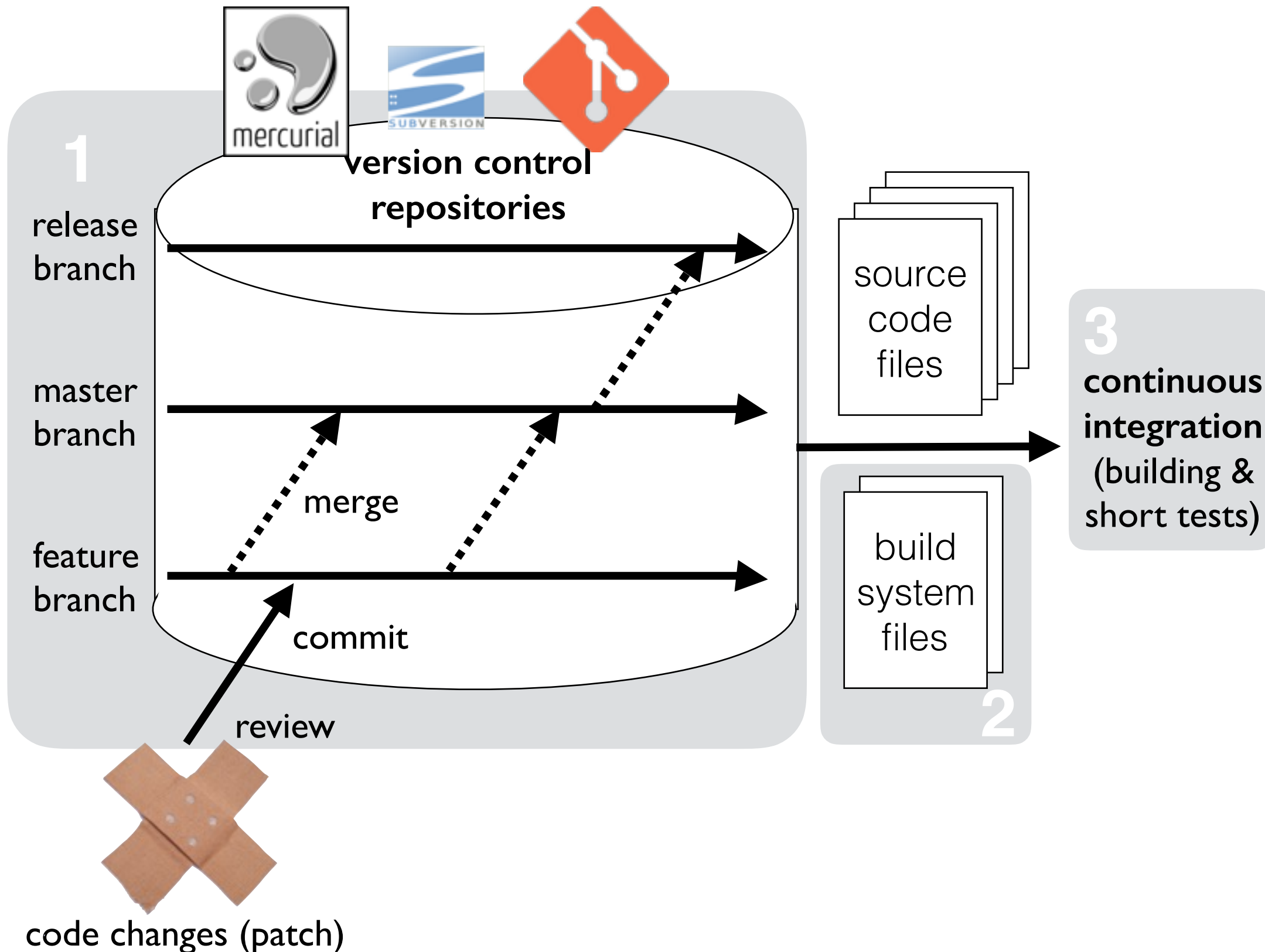
Release Engineering Pipeline (I)



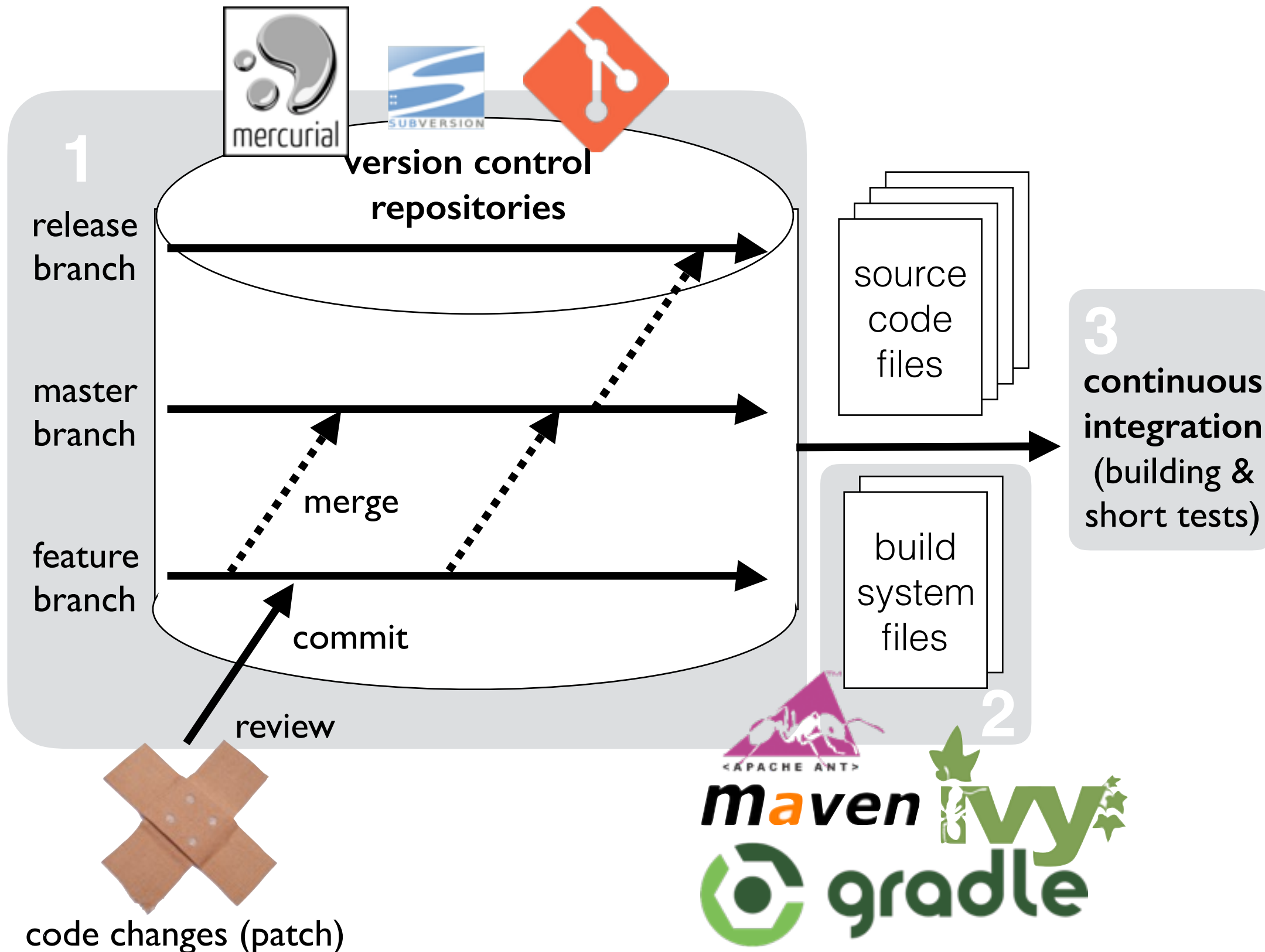
Release Engineering Pipeline (I)



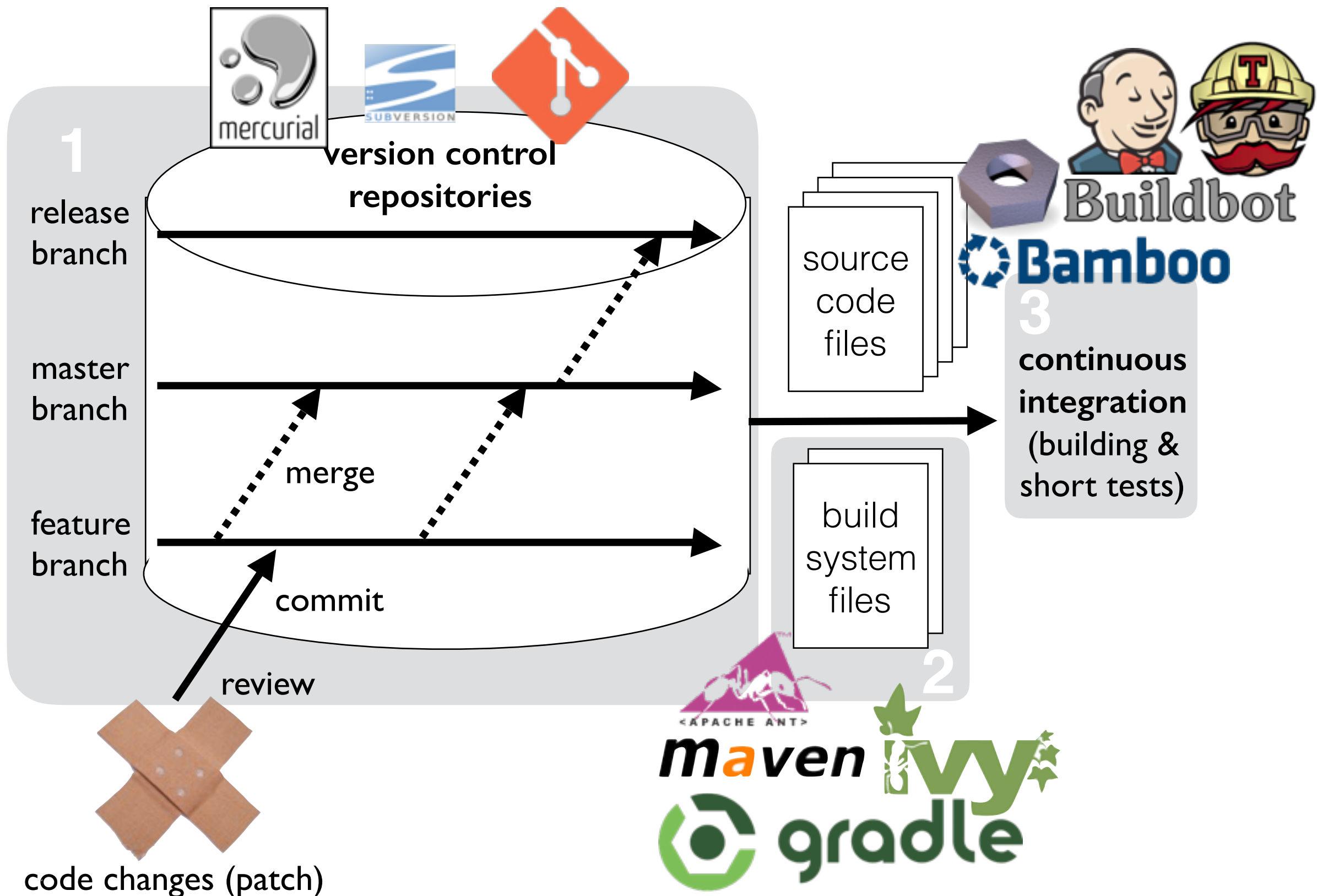
Release Engineering Pipeline (I)




Release Engineering Pipeline (I)





Release Engineering Pipeline (I)





Search all repositories

My Repositories +

 **bramadams/moosetracks** # 8

 Duration: 21 sec

 Finished: 5 months ago

bramadams / moosetracks 


build failing





































Current

Branches

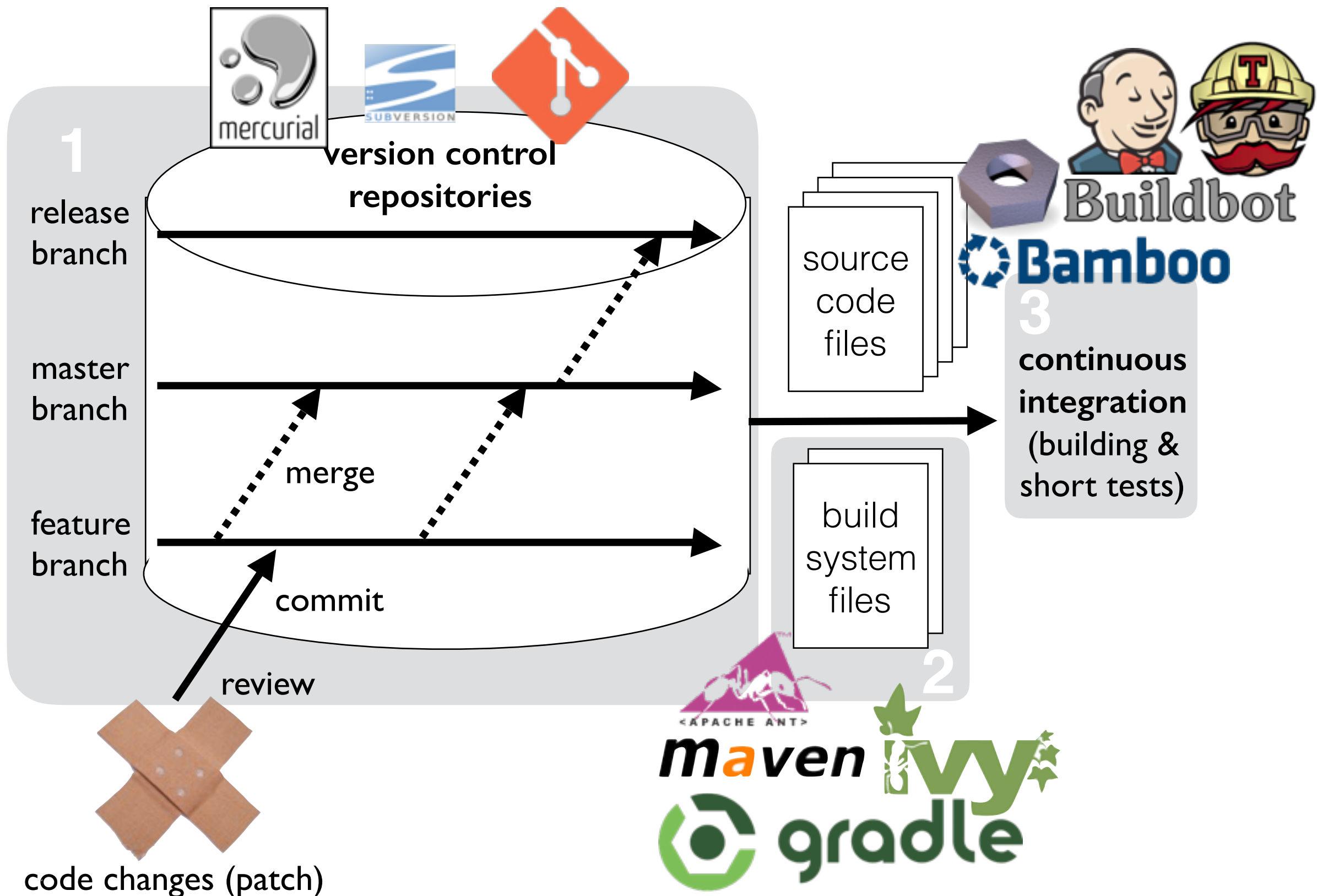
Build History

Pull Requests

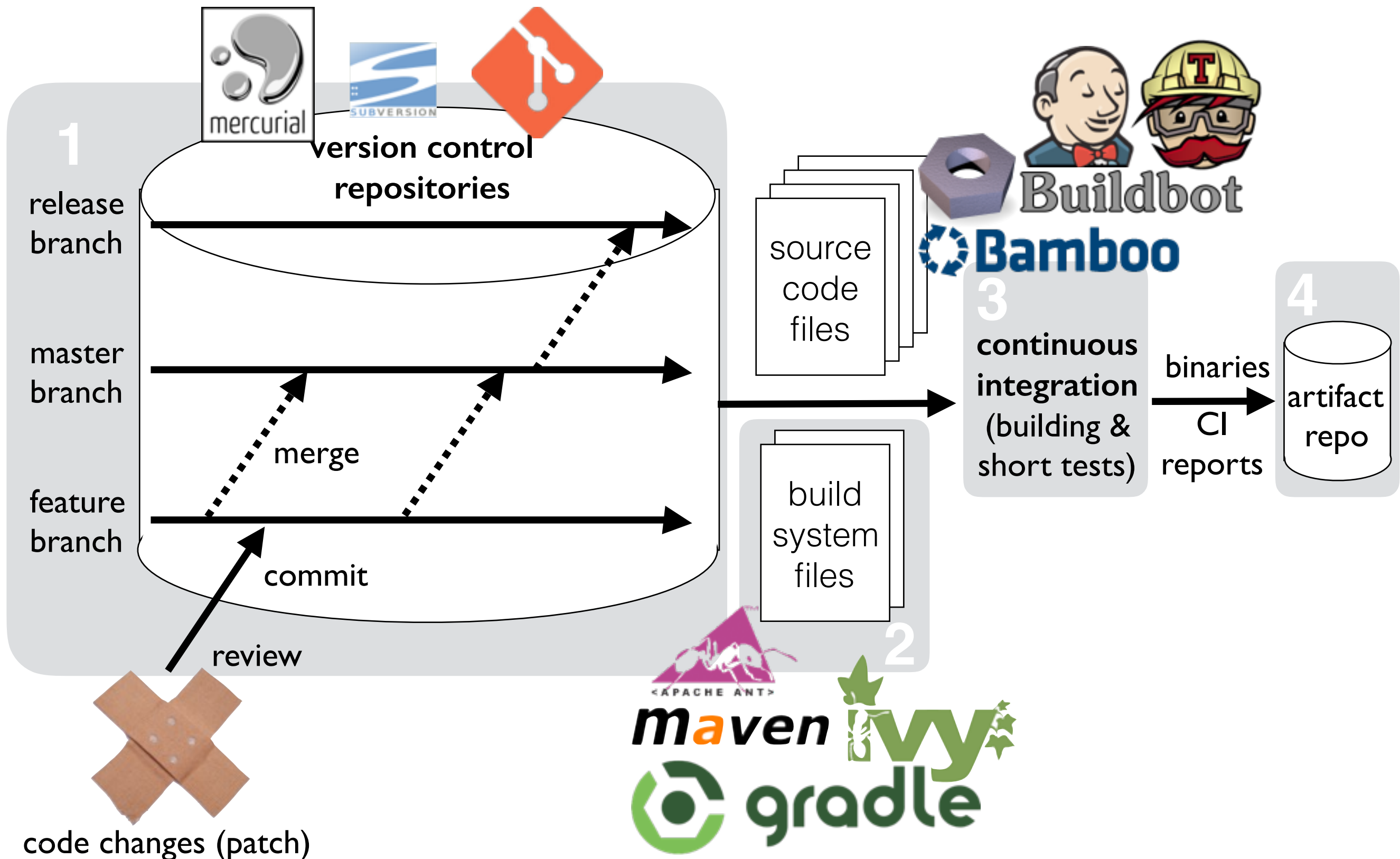
More options

 master  bramadams	Getting rid of setenv.build script.	 #8 failed  4d30f77	 21 sec  5 months ago
 master  bramadams	Removing redundant stuff.	 #7 failed  3c1811d	 19 sec  5 months ago
 master  bramadams	Fixing Jenkins redirection by extra wrapper.	 #6 passed  de2b821	 22 sec  6 months ago
 master  bramadams	Small typo.	 #5 passed  77a9adb	 21 sec  6 months ago
 master  bramadams	Making Linux 32 configuration.	 #4 passed  fd597d5	 26 sec  6 months ago
 master  bramadams	Whitespace to check out tagging.	 #3 passed  977fcda	 23 sec  6 months ago

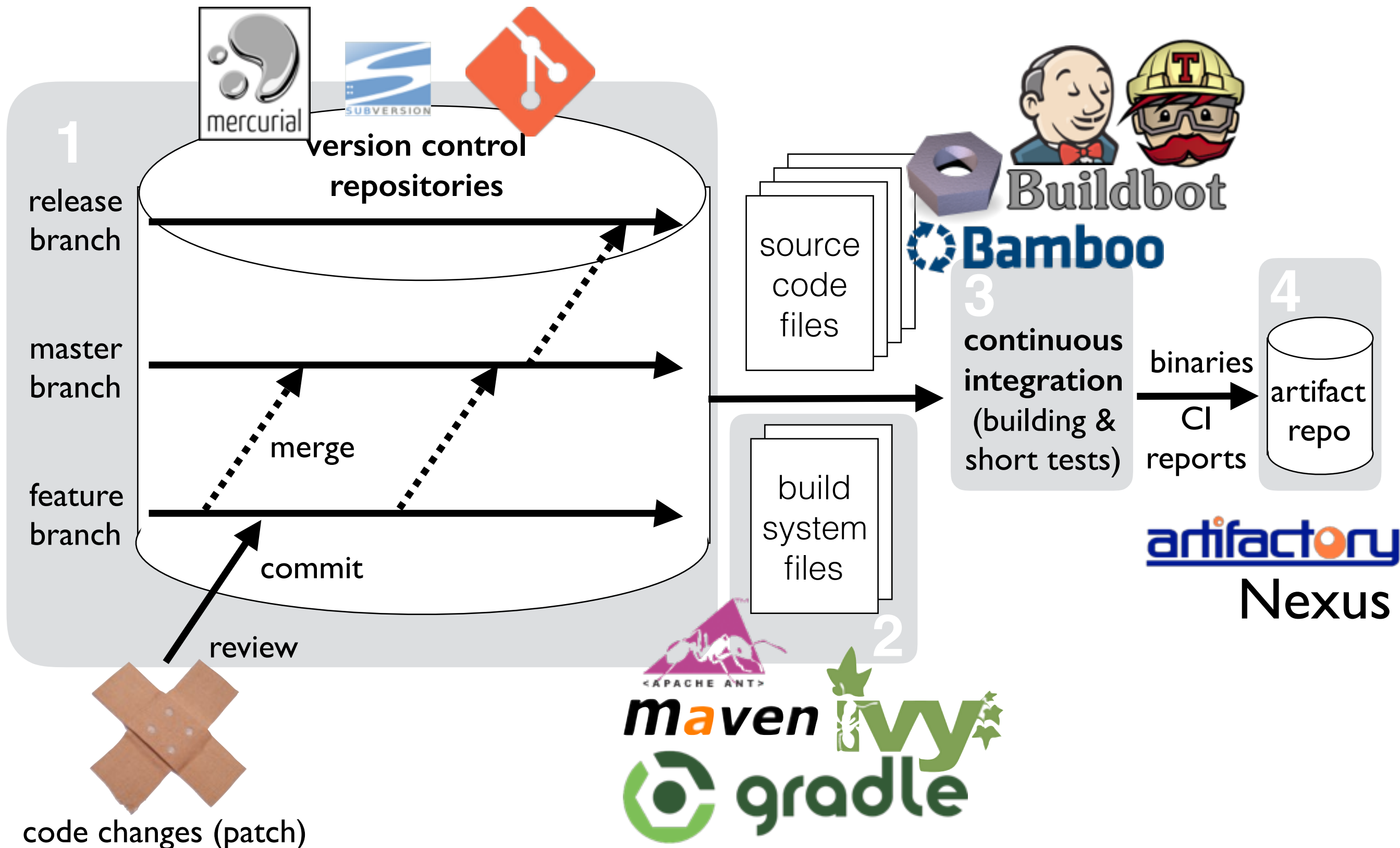
Release Engineering Pipeline (I)



Release Engineering Pipeline (I)

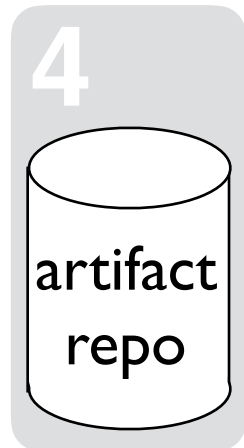


Release Engineering Pipeline (I)

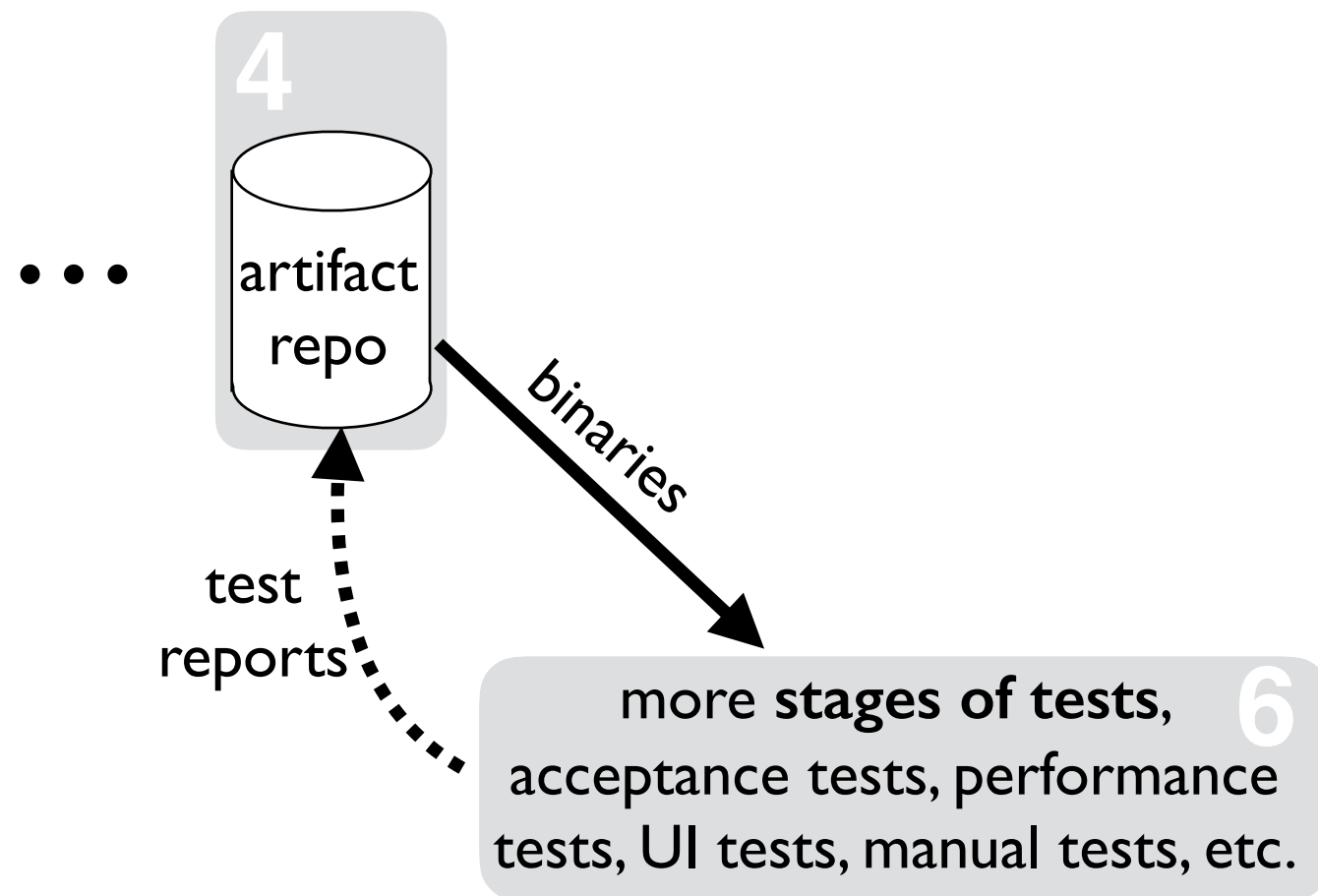


Release Engineering Pipeline (2)

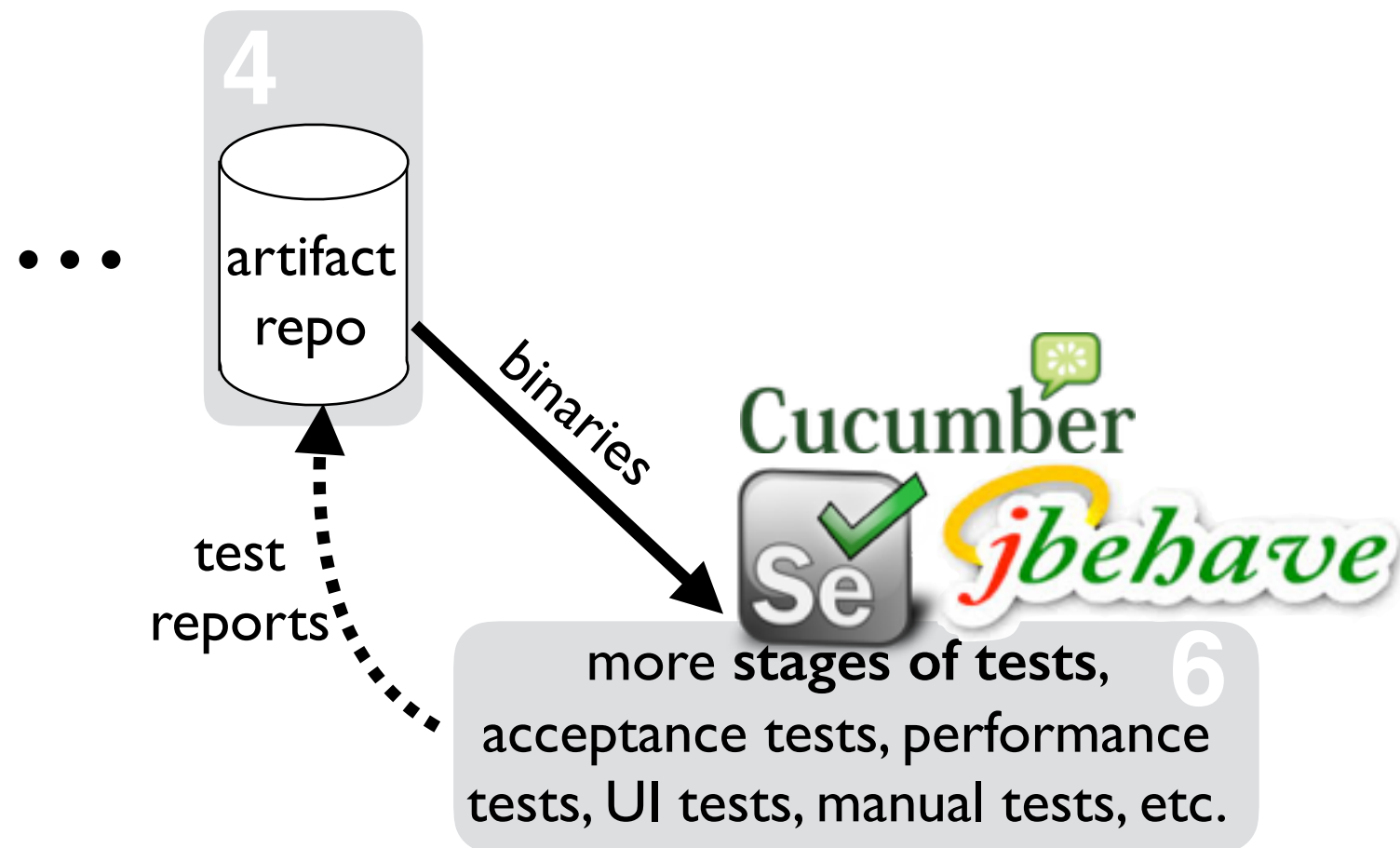
...



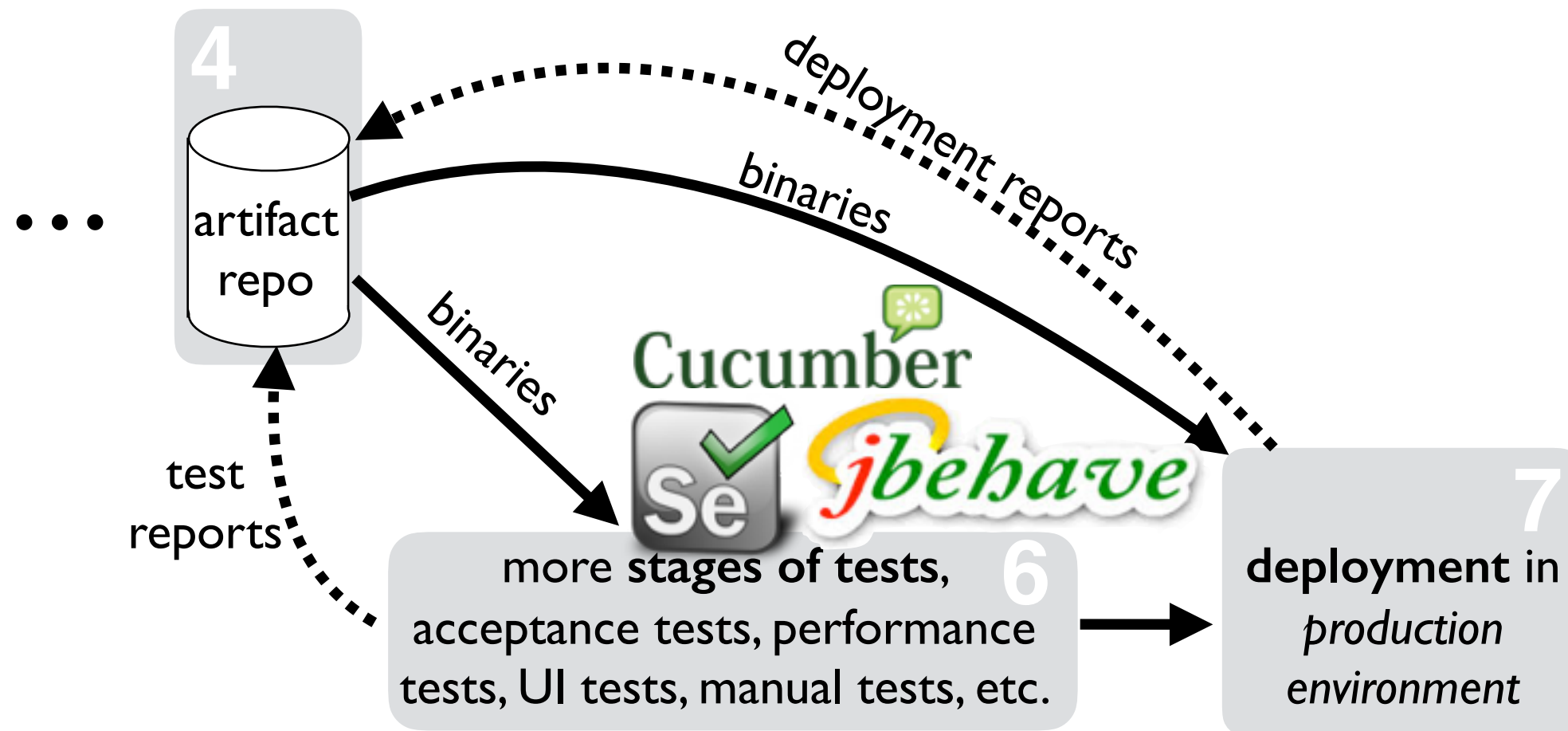
Release Engineering Pipeline (2)



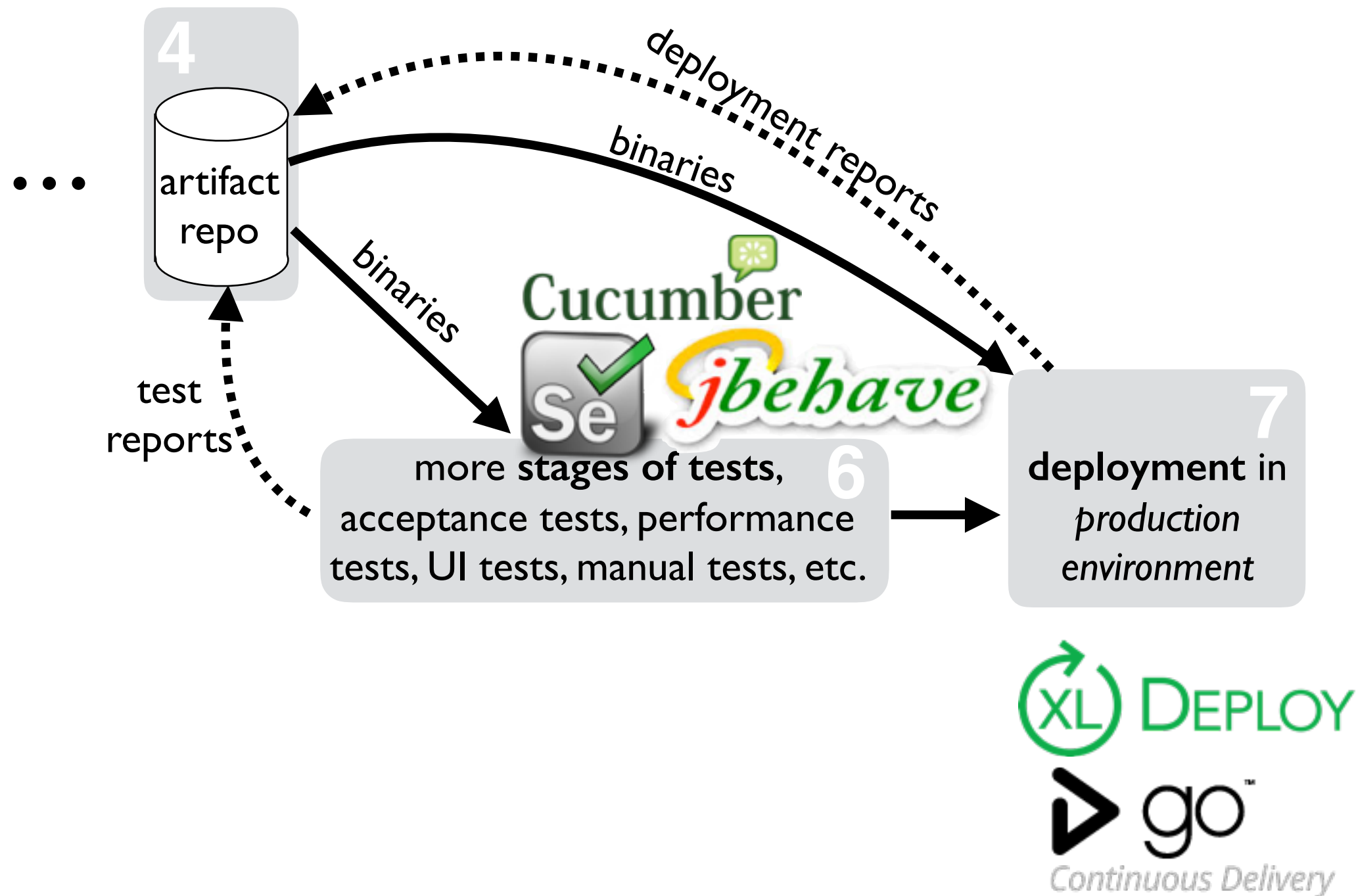
Release Engineering Pipeline (2)



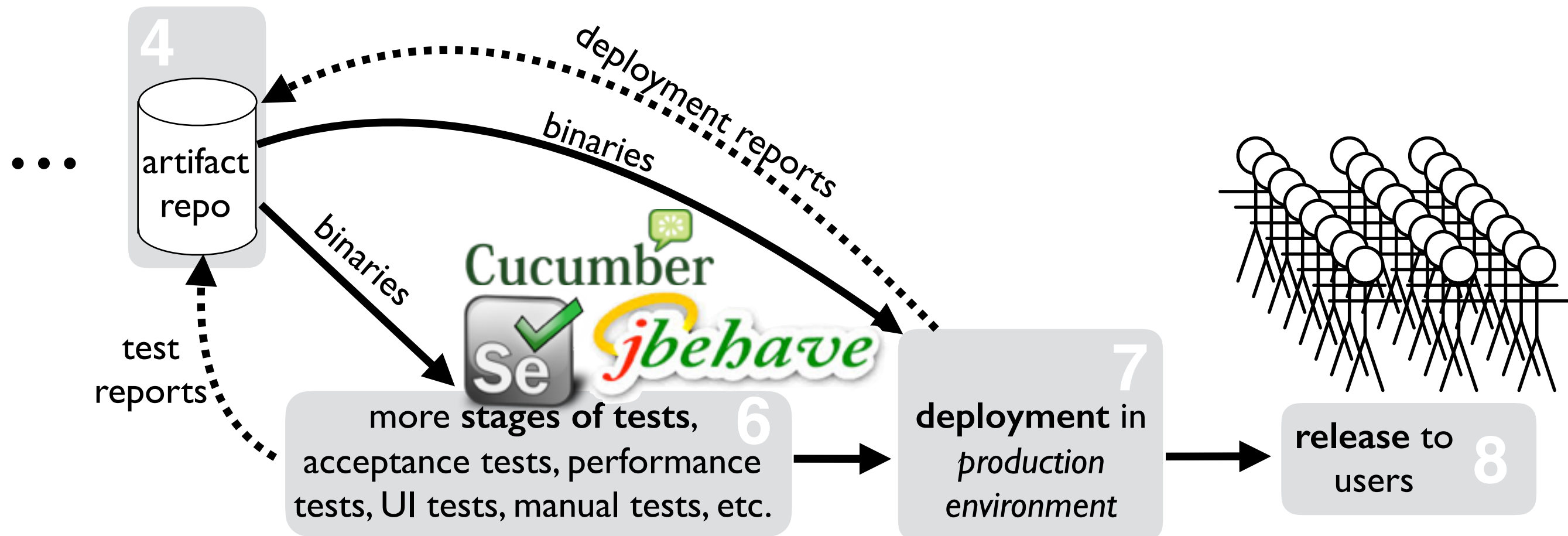
Release Engineering Pipeline (2)



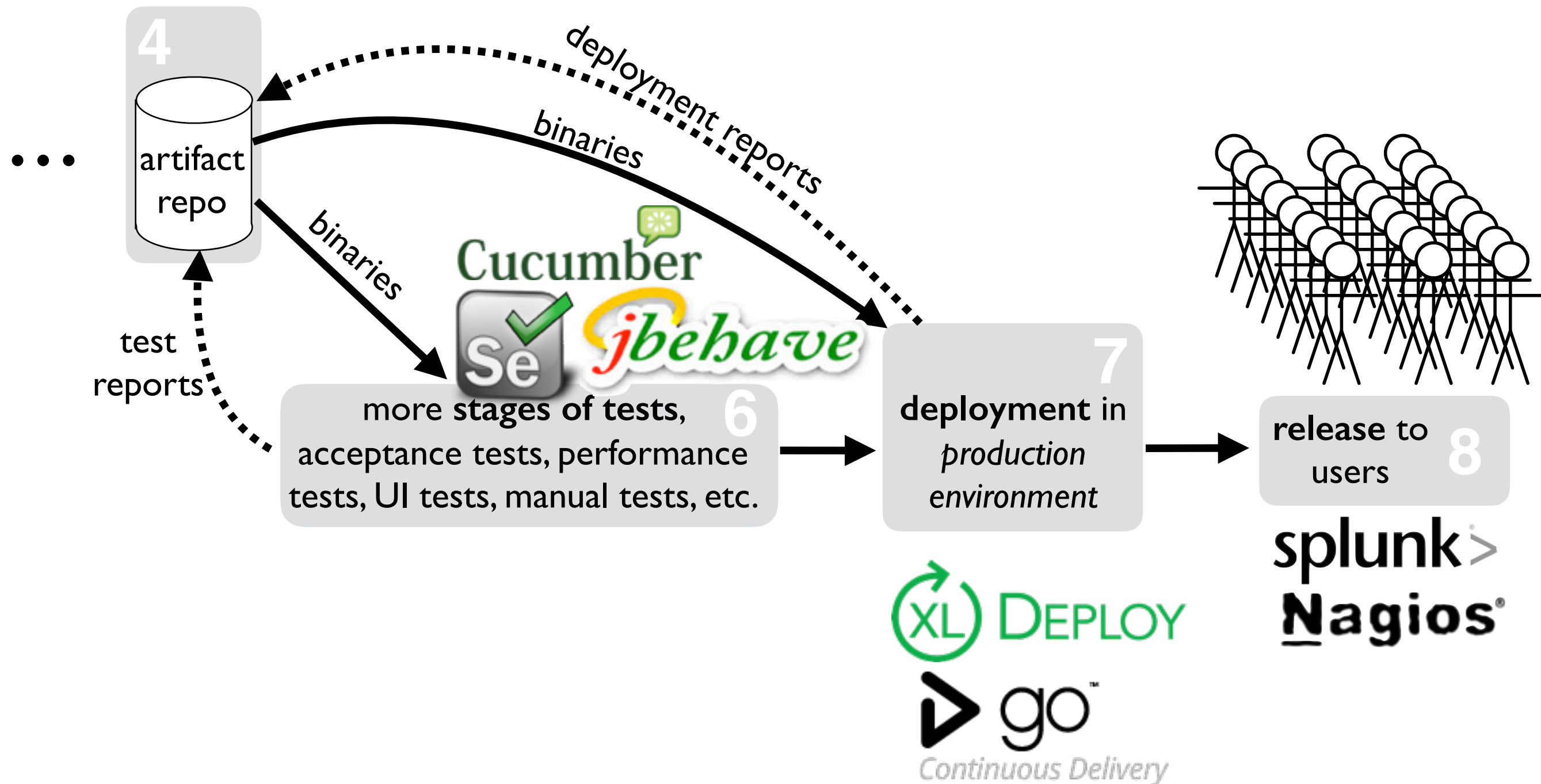
Release Engineering Pipeline (2)



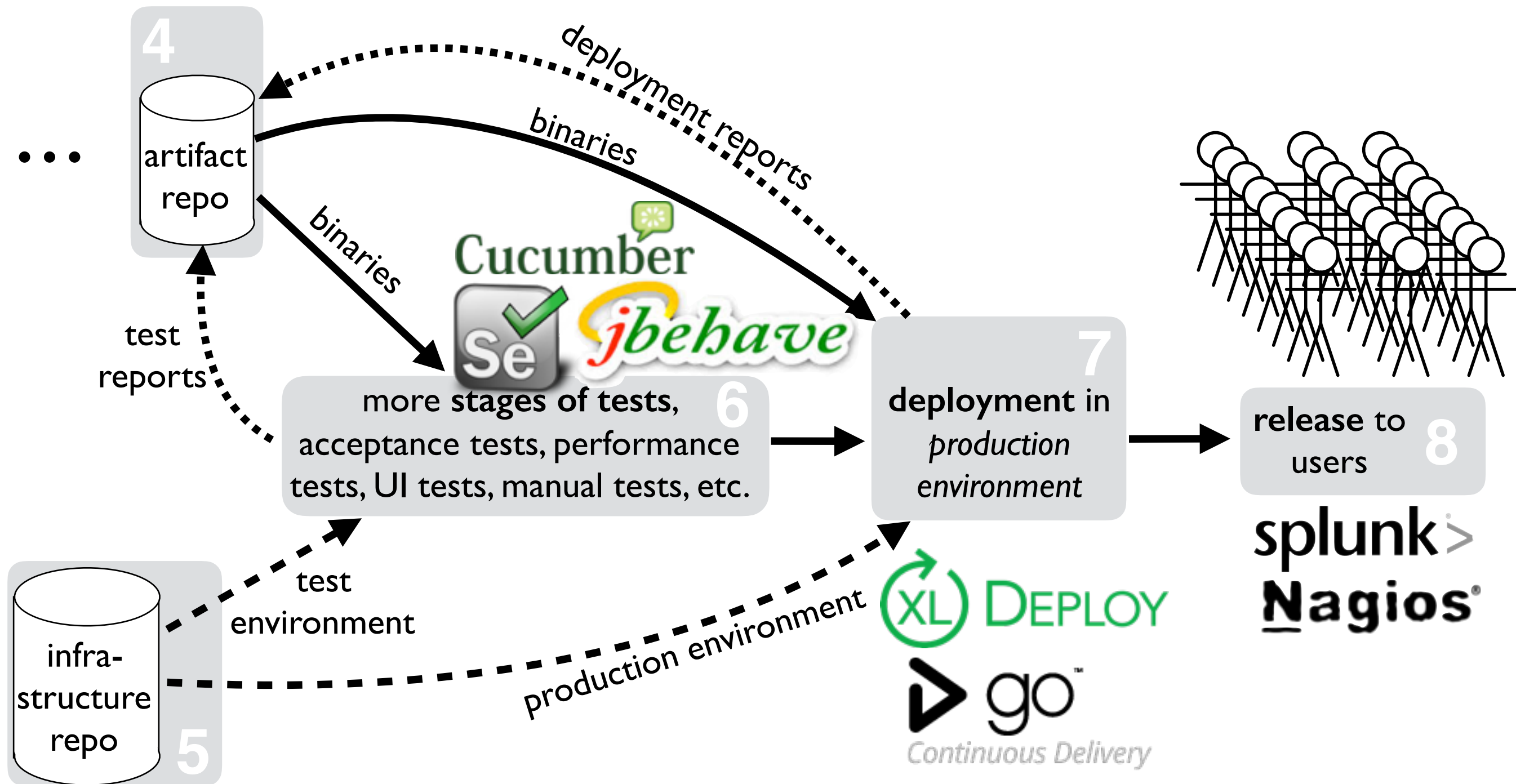
Release Engineering Pipeline (2)



Release Engineering Pipeline (2)



Release Engineering Pipeline (2)



Provisioning a PostgreSQL DB



```
# Install PostgreSQL server and client
```

```
include_recipe "postgresql::server"
```

```
include_recipe "postgresql::client"
```

```
# Make postgresql_database resource available
```

```
include_recipe "database::postgresql"
```

```
# Create database for Rails app
```

```
db = node["practicingruby"]["database"]
```

```
postgresql_database db["name"] do
```

```
  connection(
```

```
    :host      => db["host"],
```

```
    :port      => node["postgresql"]["config"]["port"],
```

```
    :username  => db["username"],
```

```
    :password  => db["password"],
```

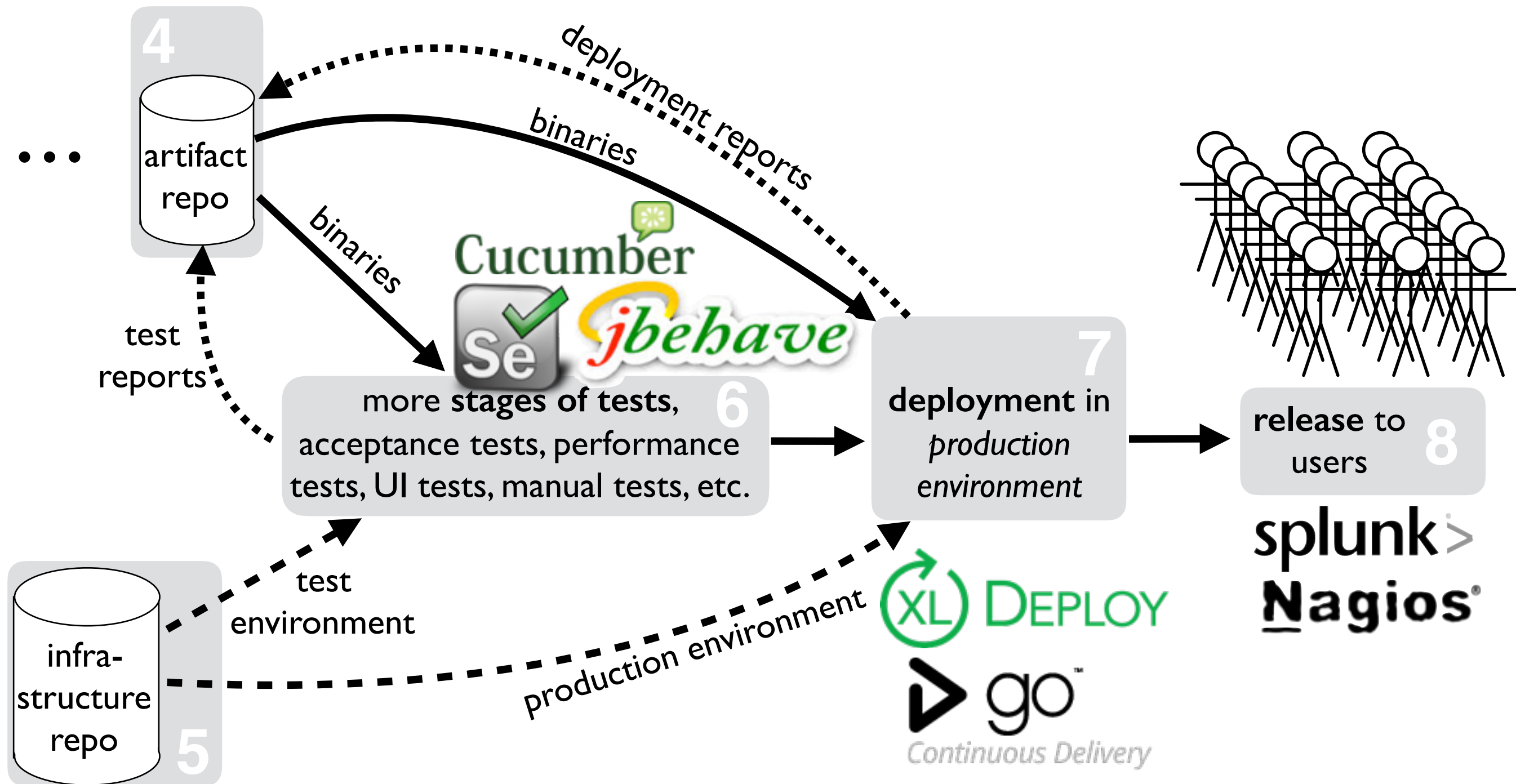
```
  )
```

```
end
```

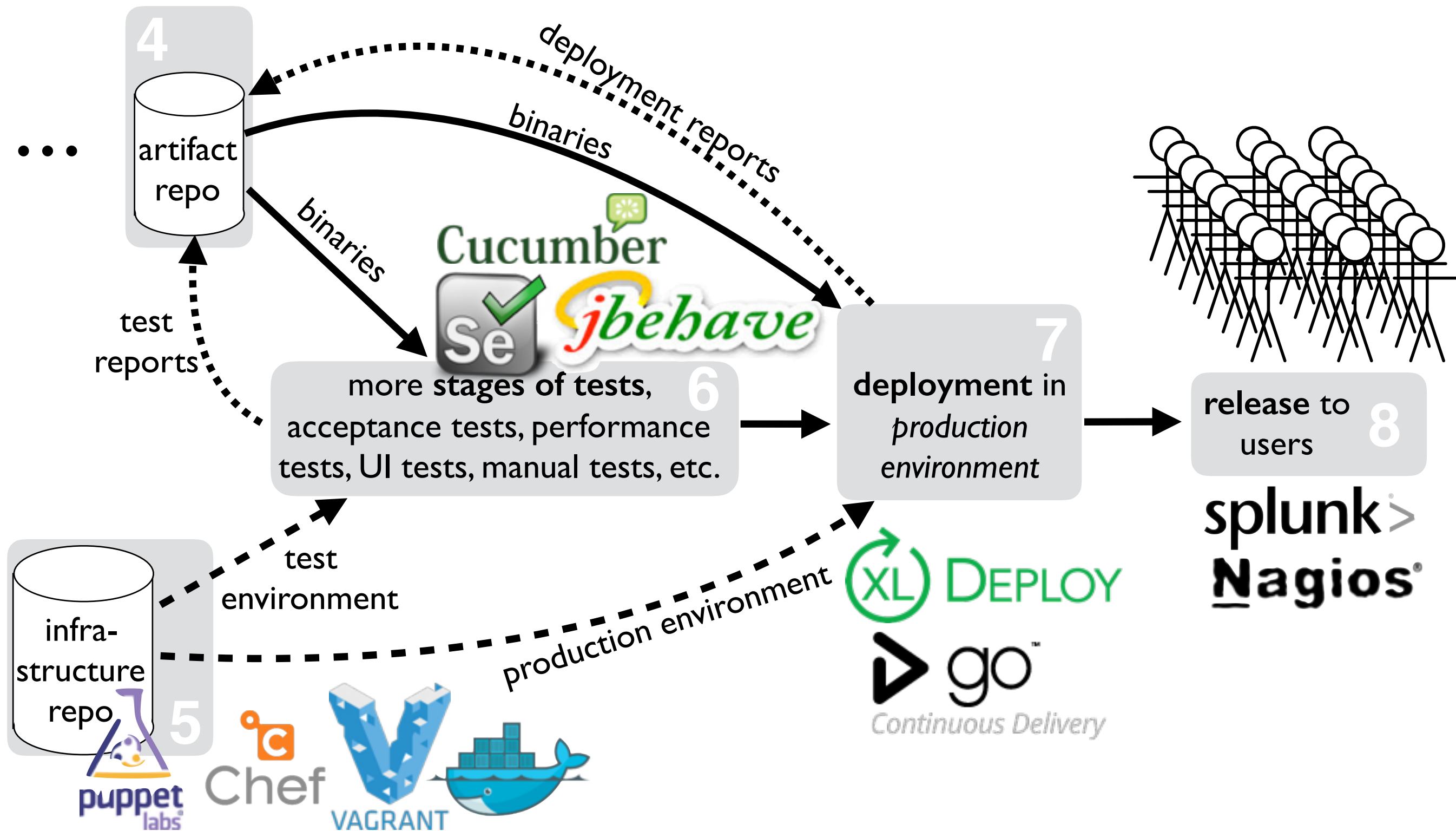
describe VM or
server
configuration in a
textual file (in VCS)

programmatic,
automatic
deployment

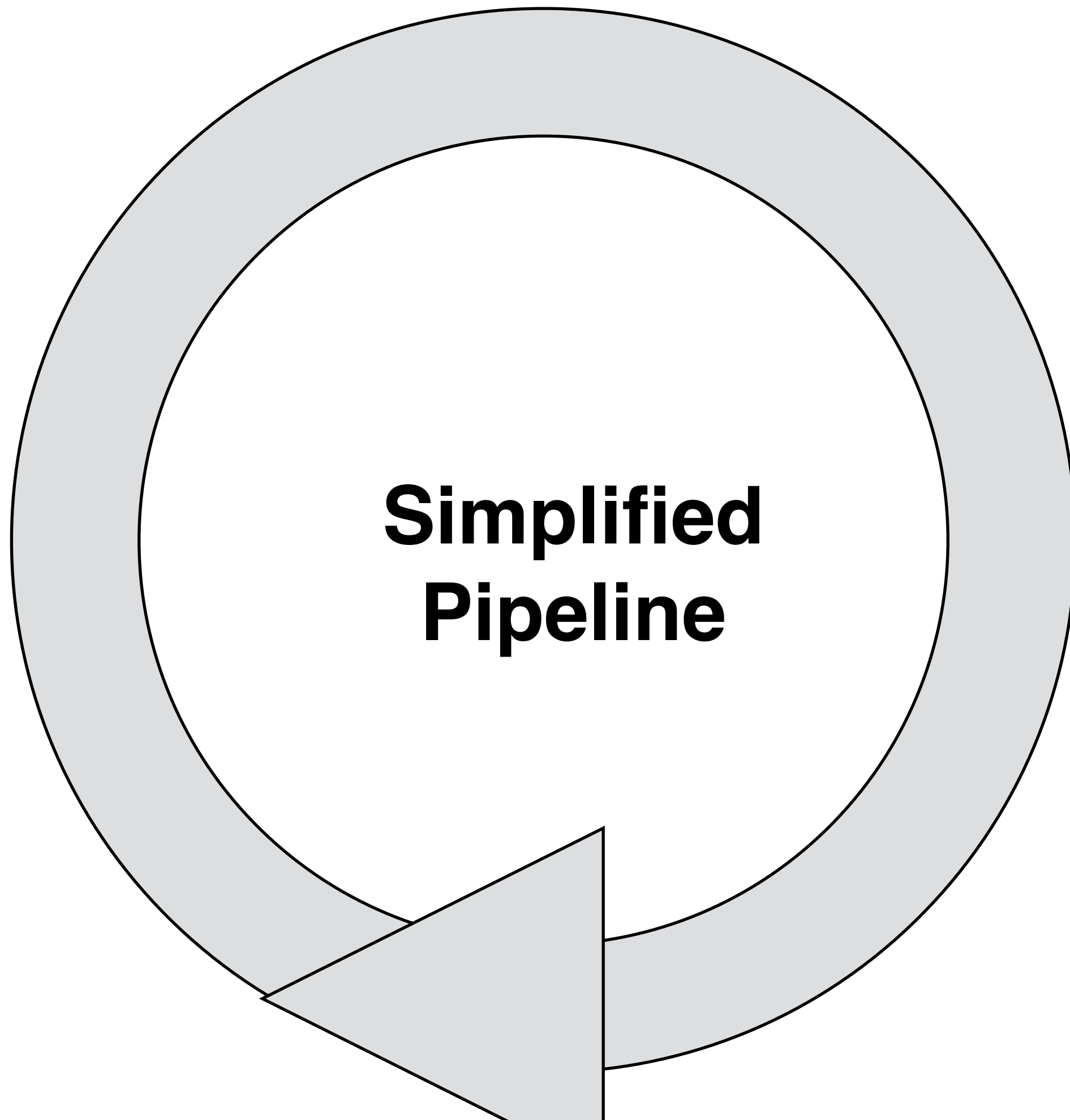
Release Engineering Pipeline (2)



Release Engineering Pipeline (2)



Simplified Pipeline



**Simplified
Pipeline**



integrating code changes

Simplified Pipeline



integrating code changes

building/testing (CI)

Simplified Pipeline





integrating code changes

building/testing (CI)

Simplified Pipeline

deploying a new release





integrating code changes

building/testing (CI)

Simplified Pipeline



releasing to the user

deploying a new release



Hold On, Isn't that Called **DevOps**?



Andrej Dyck @
RELENG '15

A. Dyck, R. Penners, and H. Lichter. Towards definitions for release engineering and devops, RELENG '15
N. Kerzazi and B. Adams. Who Needs Release and DevOps Engineers, and Why?, CSED '16



Andrej Dyck @
RELENG '15

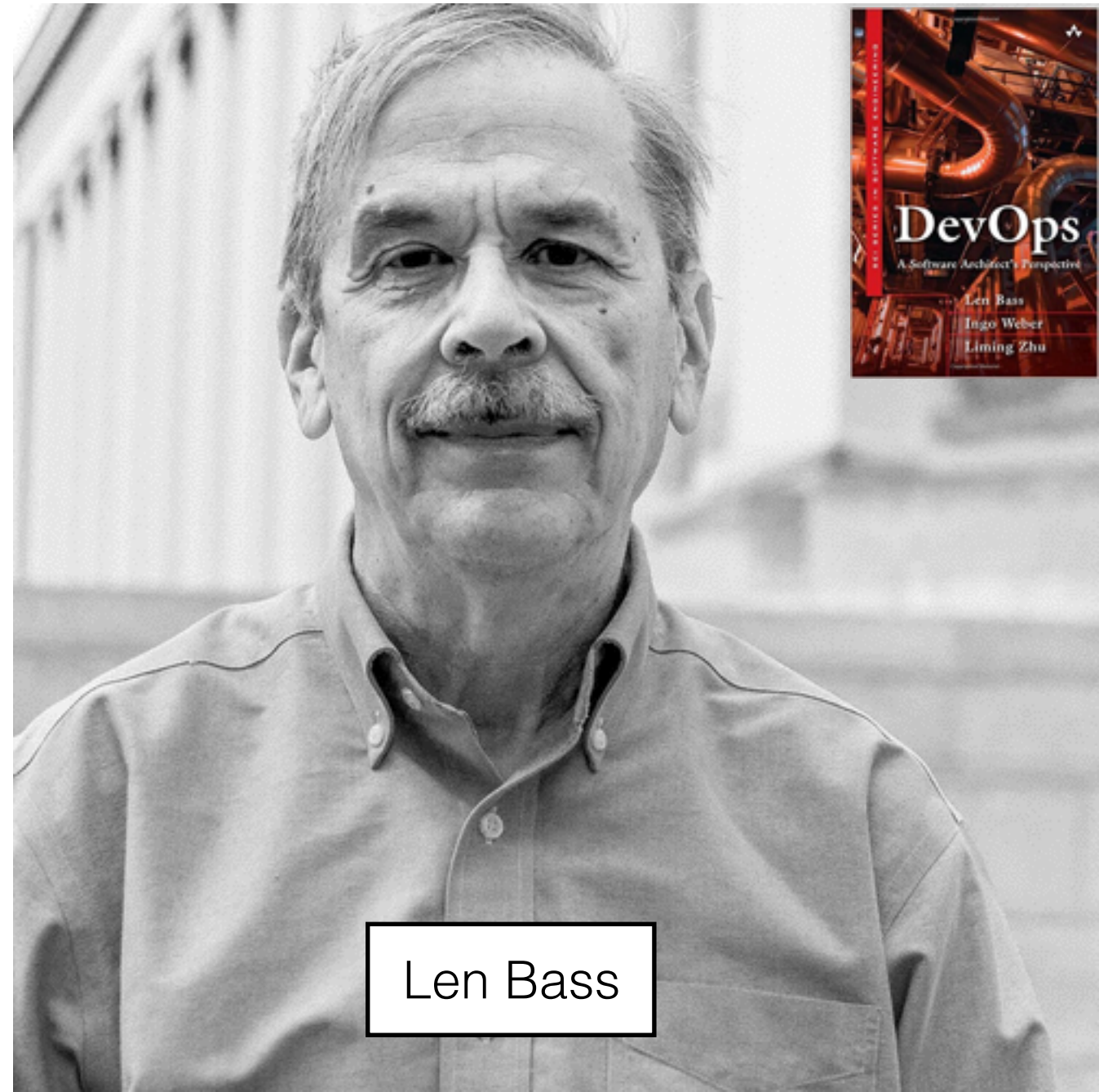
Release engineering is a software engineering **discipline** concerned with the development, implementation, and improvement of processes to **deploy high-quality software reliably and predictably**.



Andrej Dyck @
RELENG '15

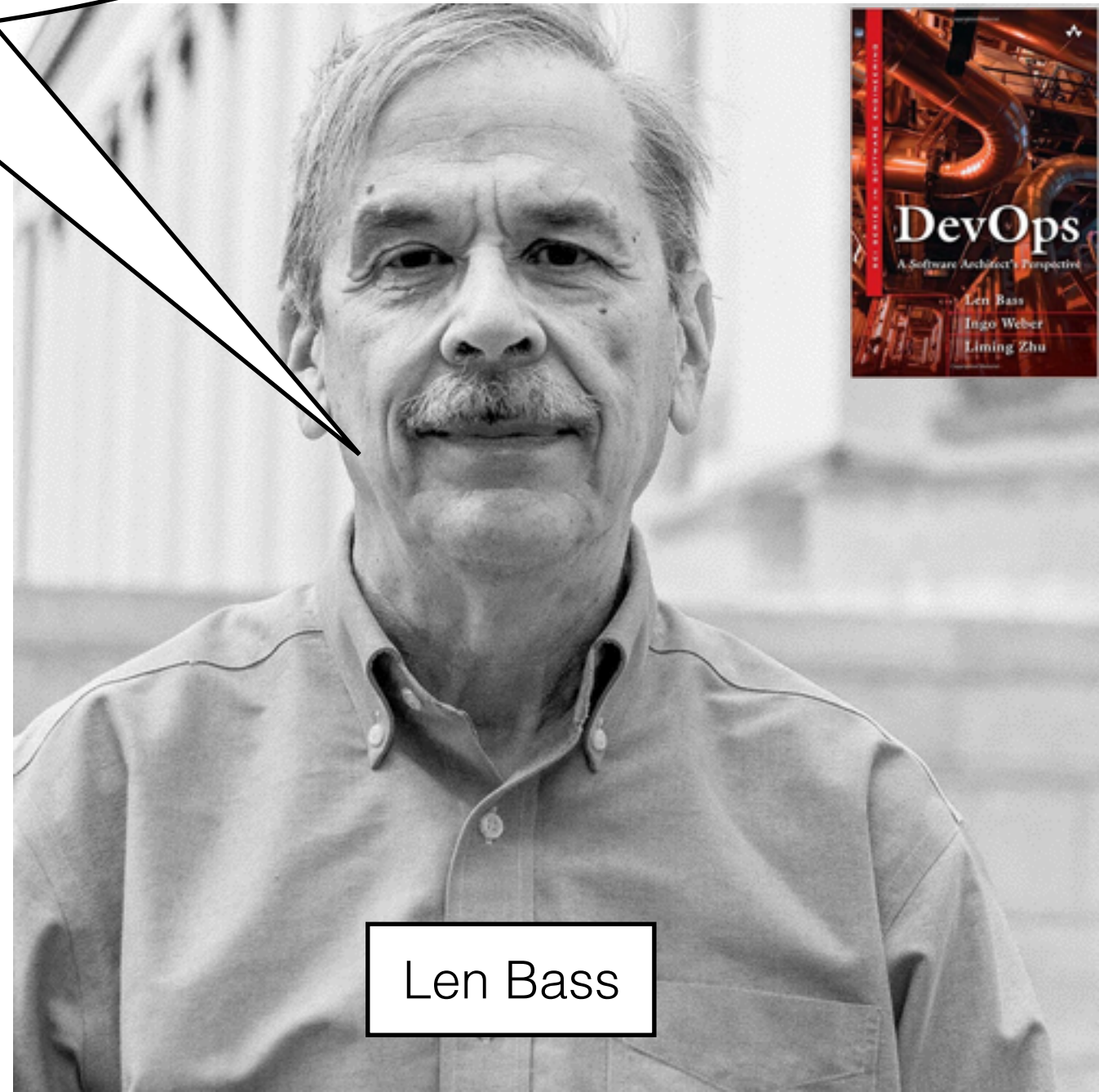
Release engineering is a software engineering **discipline** concerned with the development, implementation, and improvement of processes to **deploy high-quality software reliably and predictably**.

DevOps is an **organizational approach** that stresses empathy and cross-functional **collaboration within and between** teams – especially **development and IT operations** – in software development organizations, in order to operate resilient systems and accelerate delivery of changes.



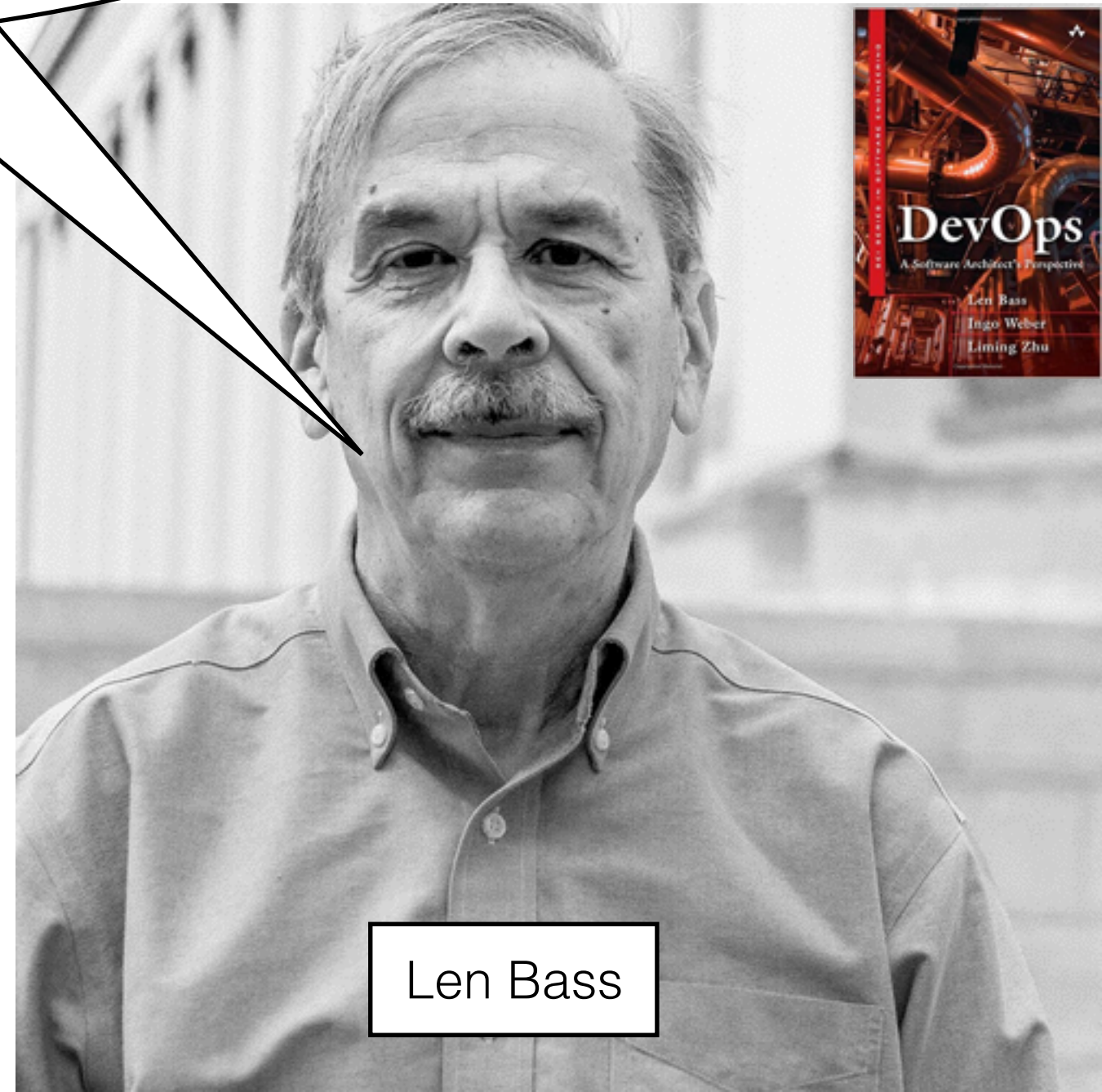
Len Bass

DevOps is a **set of practices** intended to reduce the time between committing a change to a system and the change being placed into normal production, while ensuring high quality.



Len Bass

DevOps is a **set of practices** intended to
speed up release engineering?



Len Bass

What can **you**
do for release
engineering?



integrating code changes

building/testing (CI)



releasing to the user

deploying a new release



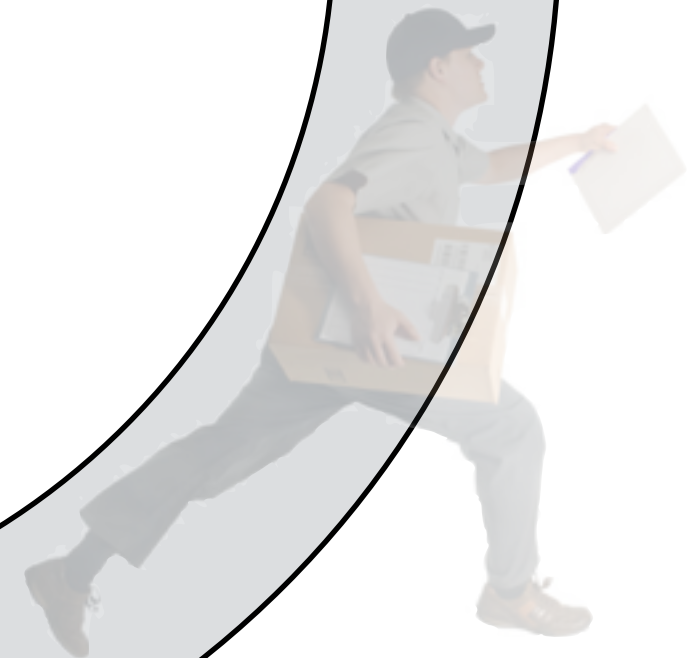
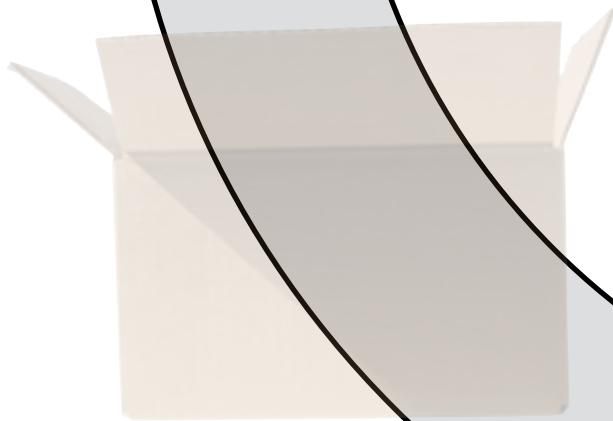


integrating code changes

building/testing (CI)

releasing to the user

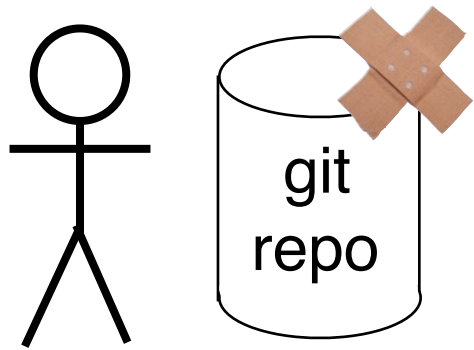
deploying a new release



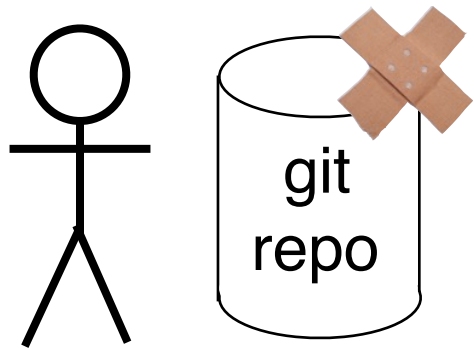
Branch-based Integration Hell



Branch-based Integration Hell



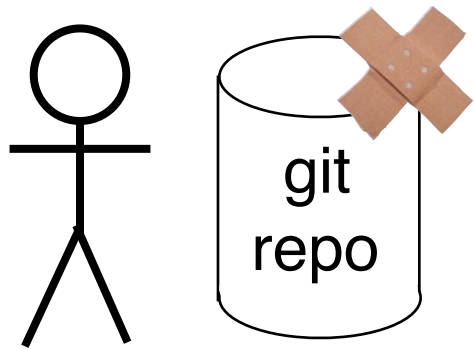
Branch-based Integration Hell



myapp
v.1



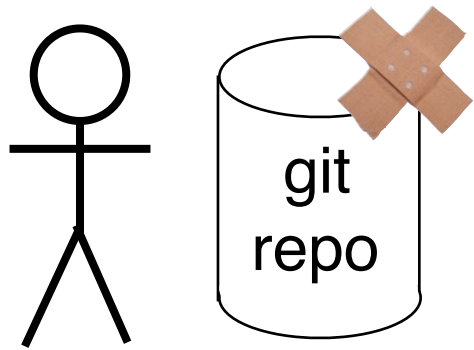
Branch-based Integration Hell



myapp
v.1



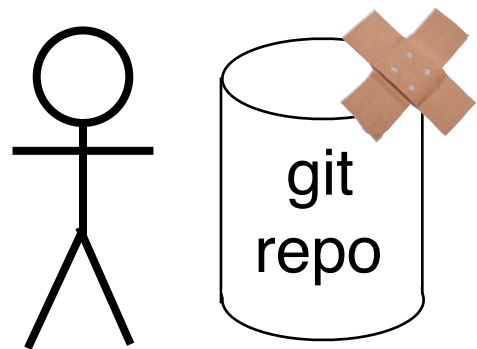
Branch-based Integration Hell



myapp
v.1



Branch-based Integration Hell

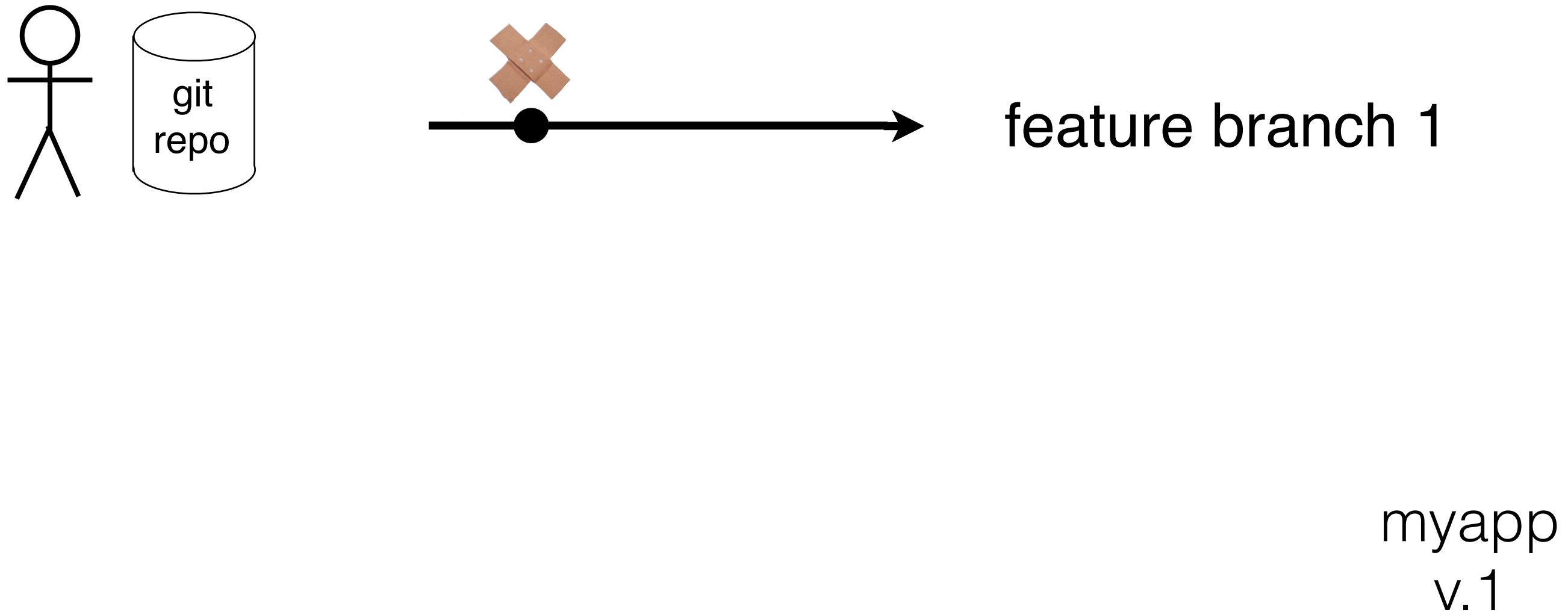


feature branch 1

myapp
v.1



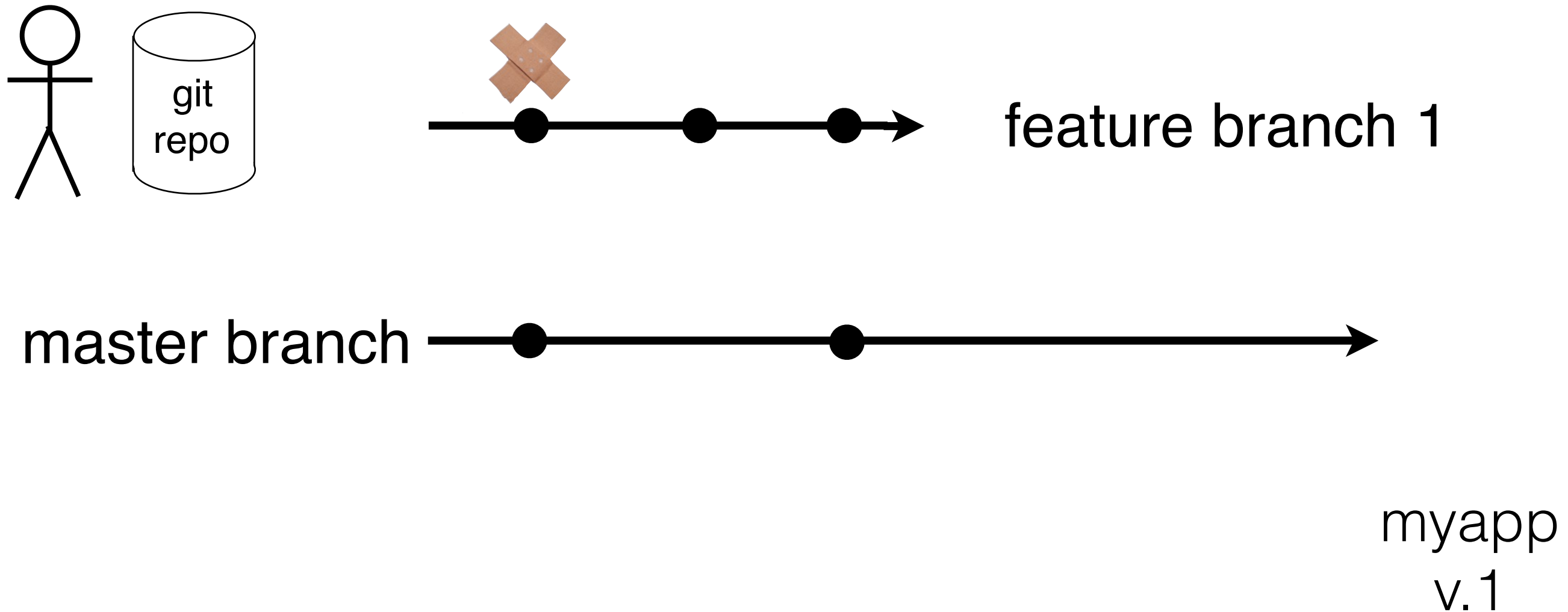
Branch-based Integration Hell



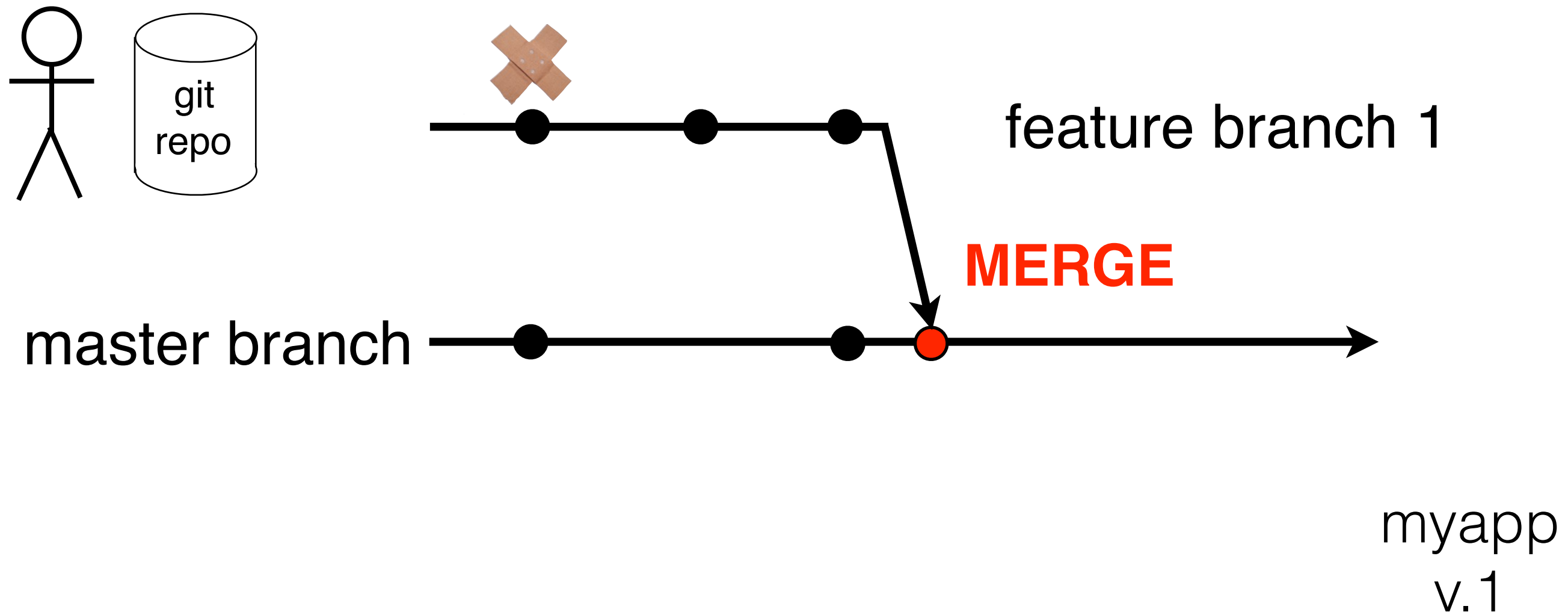
Branch-based Integration Hell



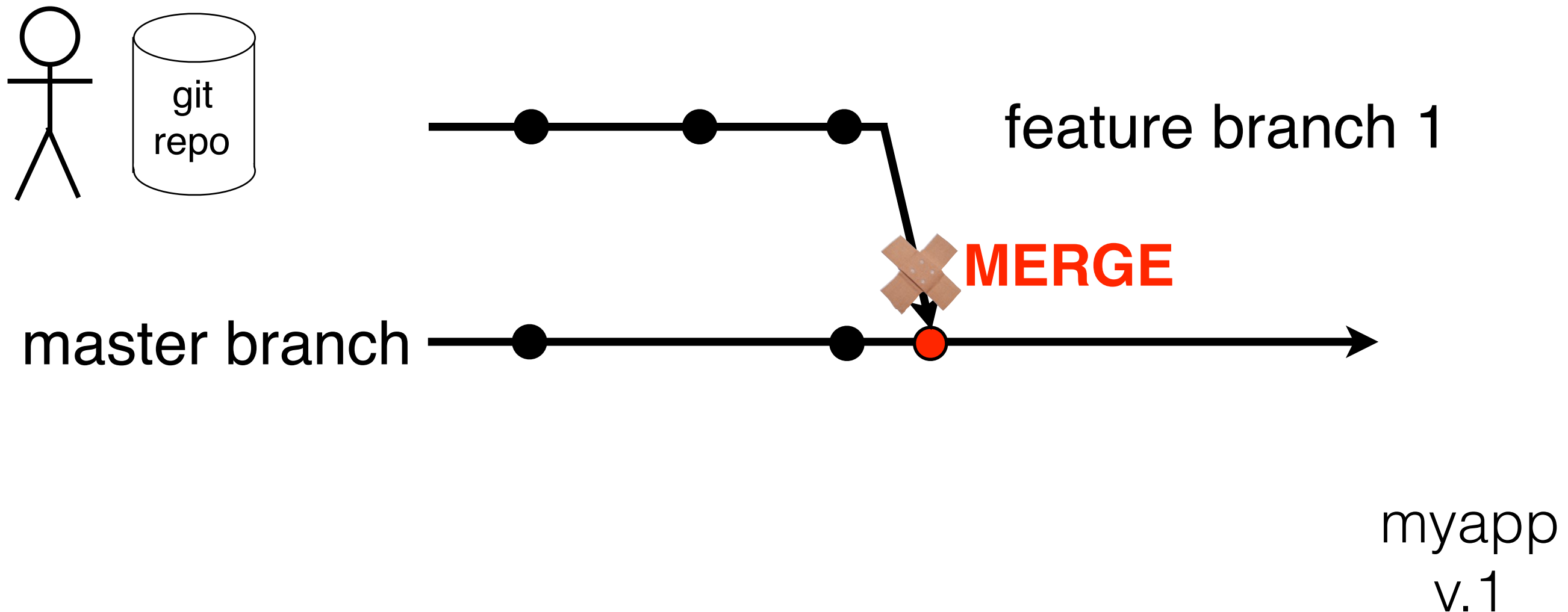
Branch-based Integration Hell



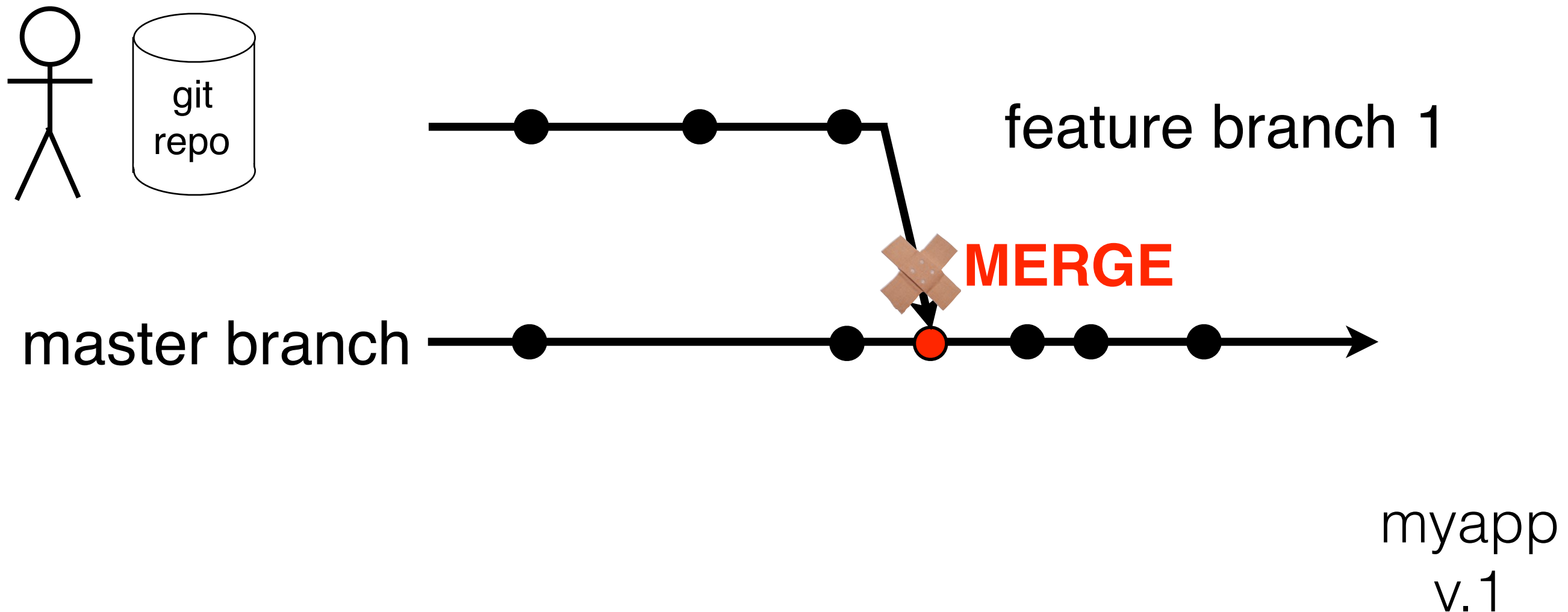
Branch-based Integration Hell



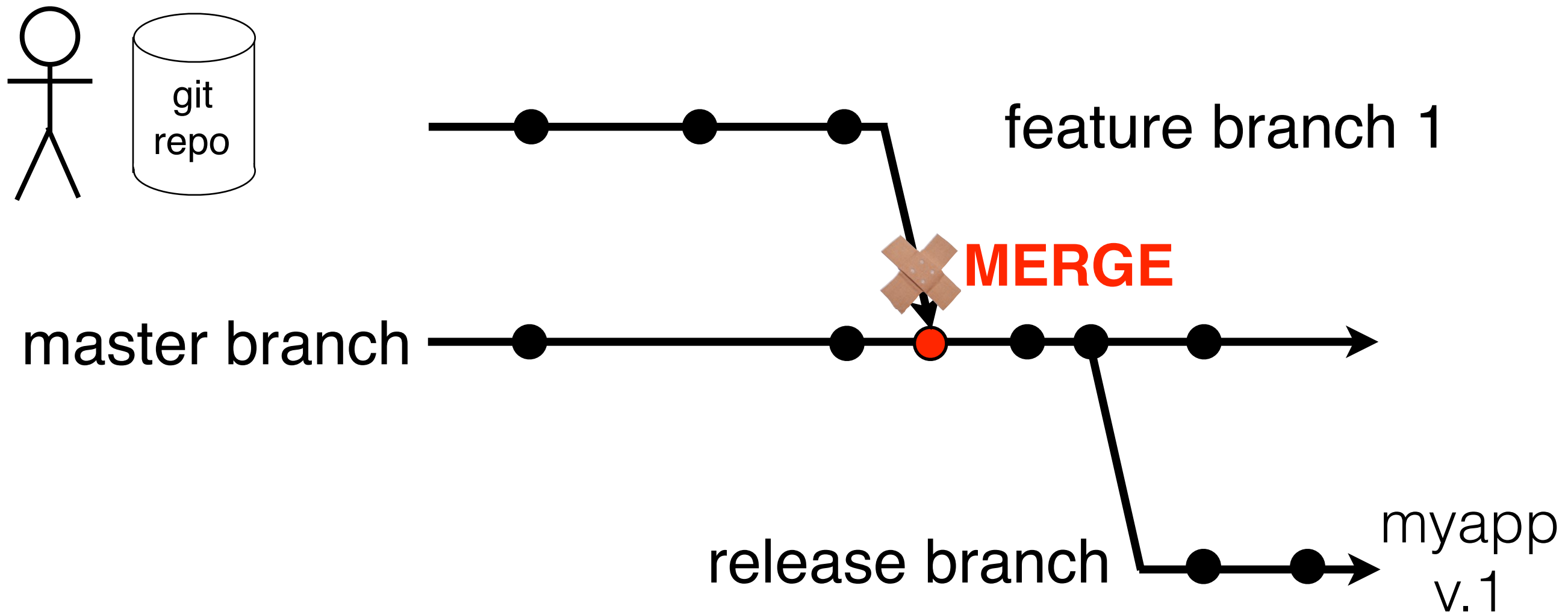
Branch-based Integration Hell



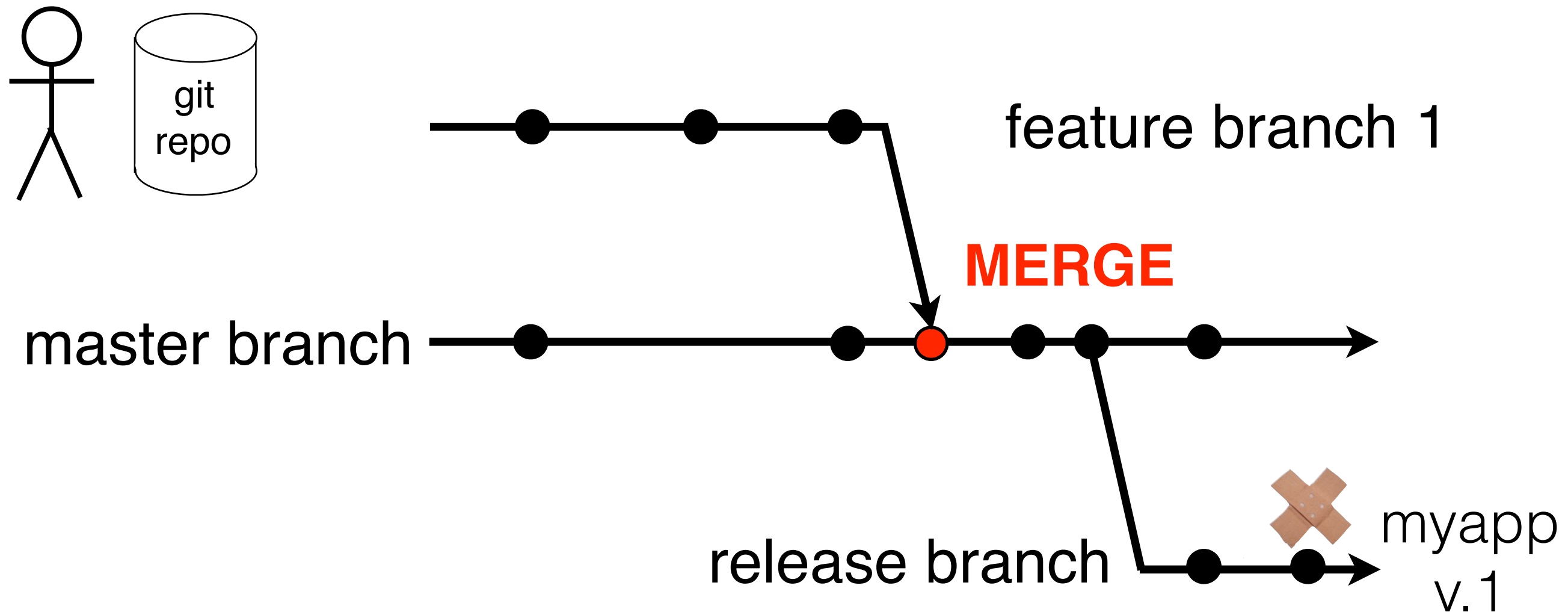
Branch-based Integration Hell



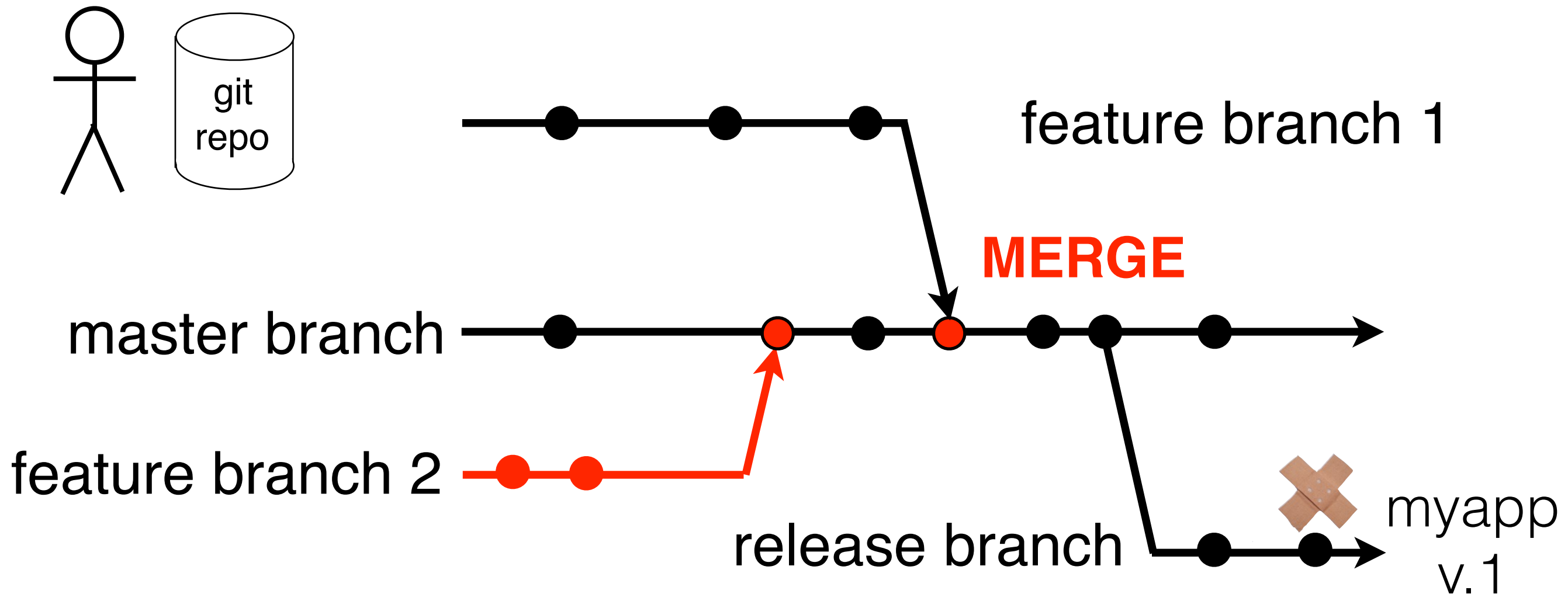
Branch-based Integration Hell



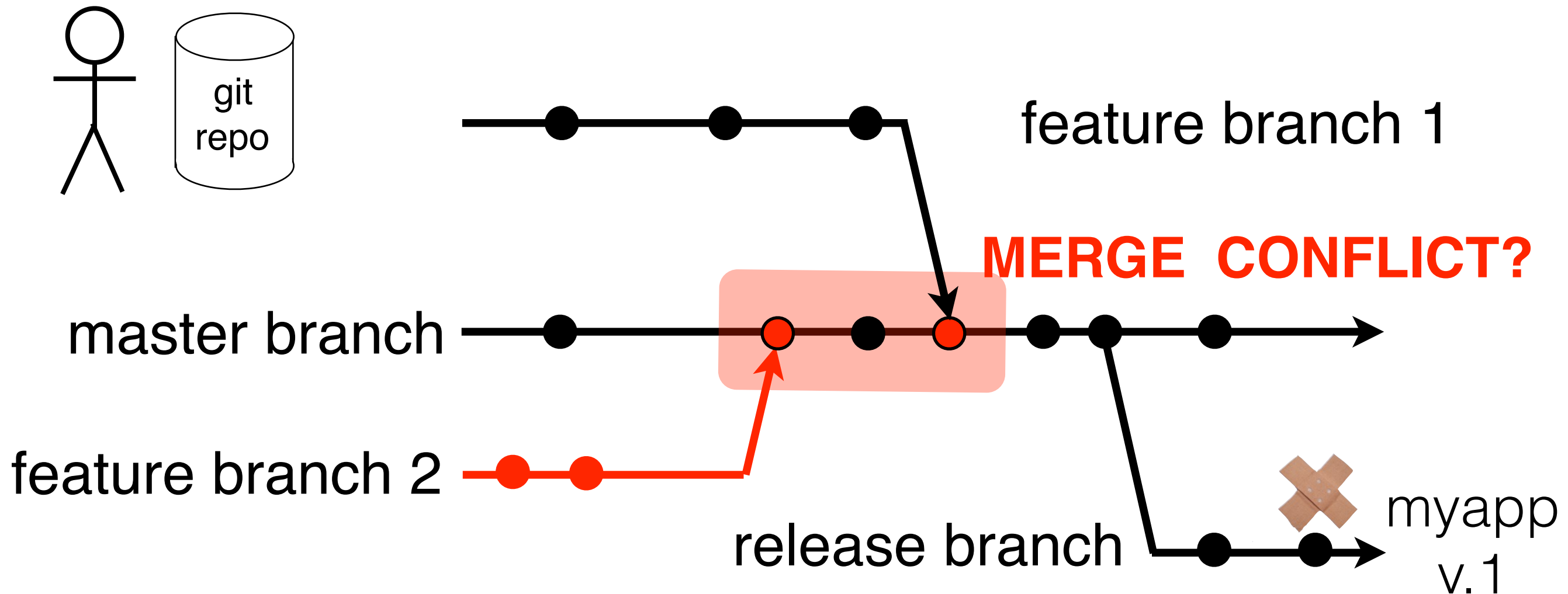
Branch-based Integration Hell



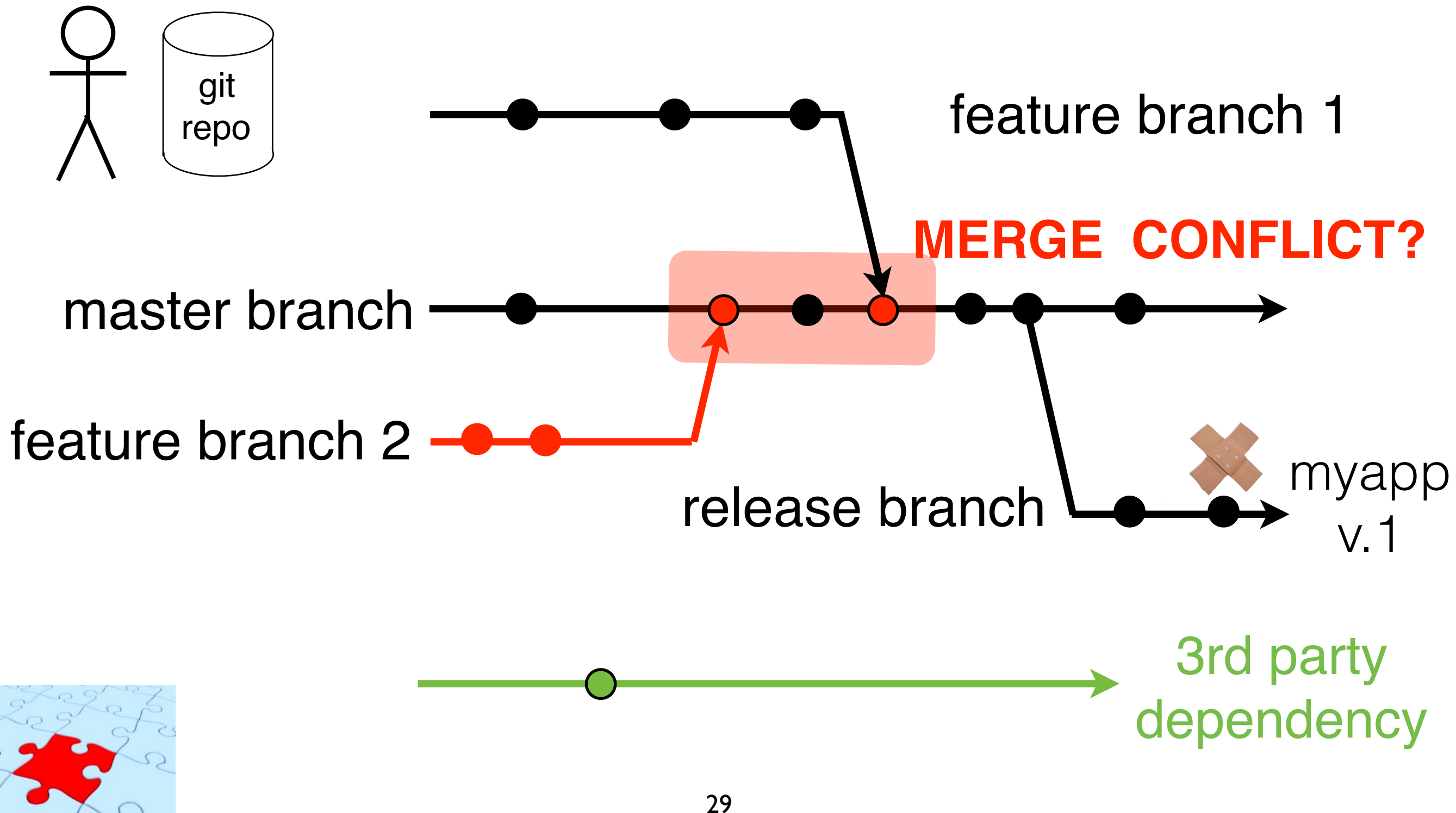
Branch-based Integration Hell



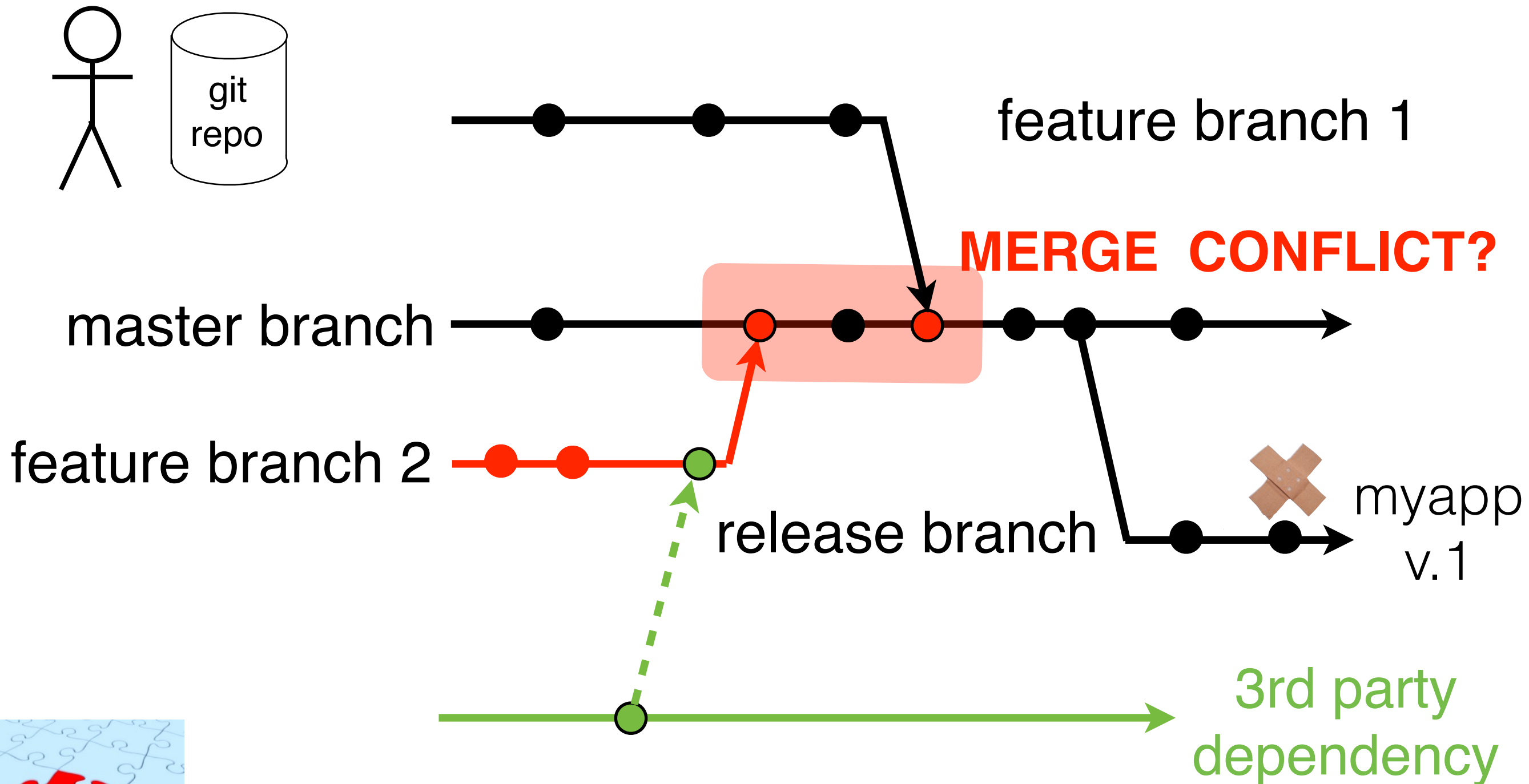
Branch-based Integration Hell



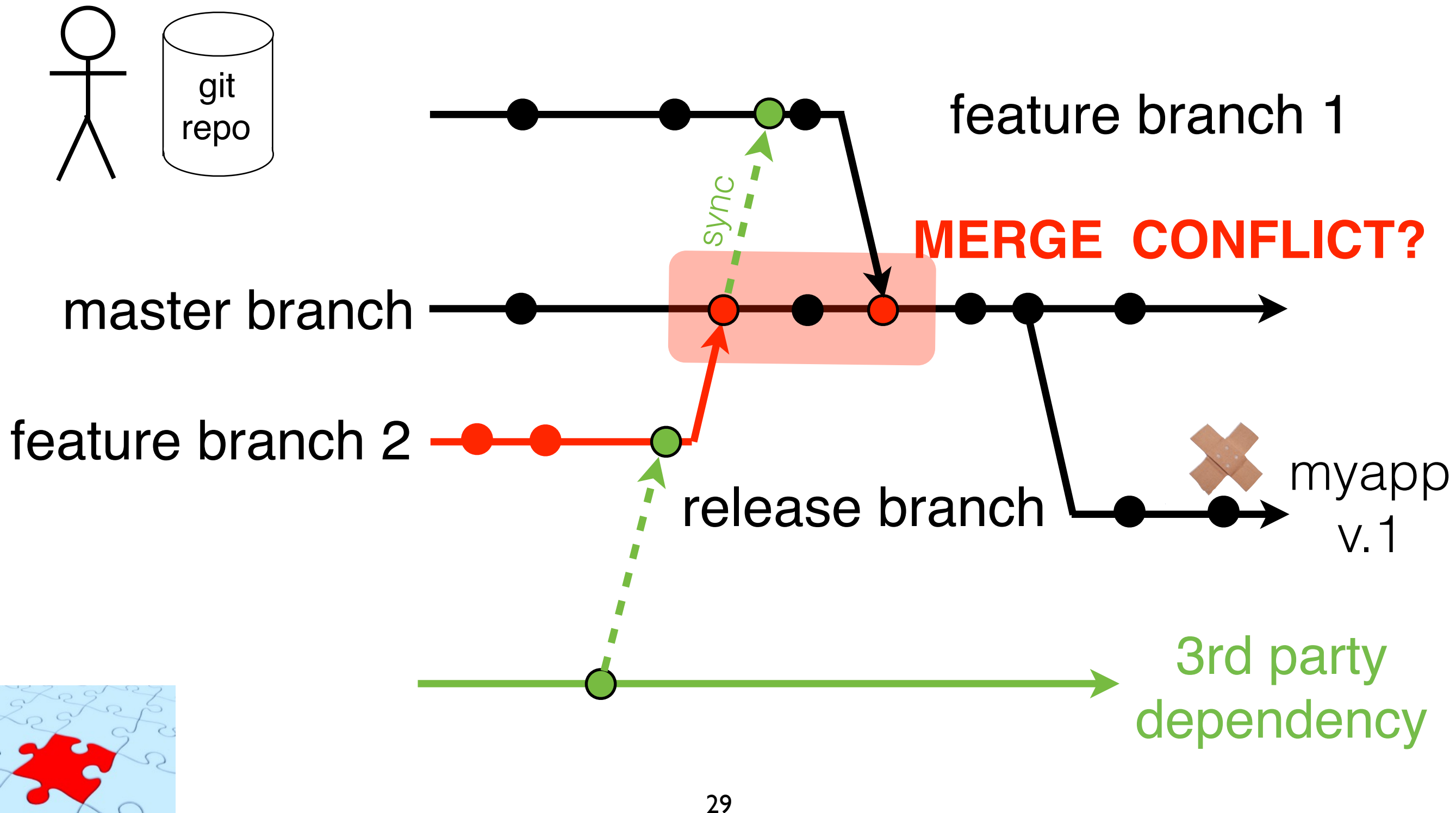
Branch-based Integration Hell



Branch-based Integration Hell



Branch-based Integration Hell



Conflicts Cause Trouble



Conflicts Cause Trouble



Conflicts Cause Trouble

on average 16% of
all merges have
textual conflicts
(manual resolution
needed)



Conflicts Cause Trouble

on average 16% of
all merges have
textual conflicts
(manual resolution
needed)

on average 5%
of all merges
have build issues

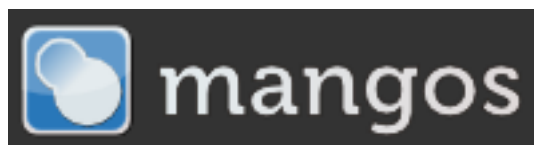


Conflicts Cause Trouble

on average 16% of
all merges have
textual conflicts
(manual resolution
needed)

on average 5%
of all merges
have build issues

on average
11.7% of all
merges have test
issues



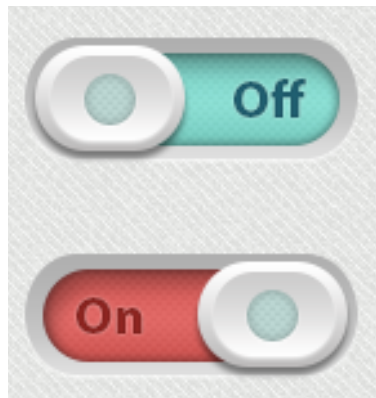
Emerging Alternative: Feature Toggles



Emerging Alternative: Feature Toggles

```
#ifdef new_feature_on  
    /* code of new feature */  
#endif
```

pre-compilation



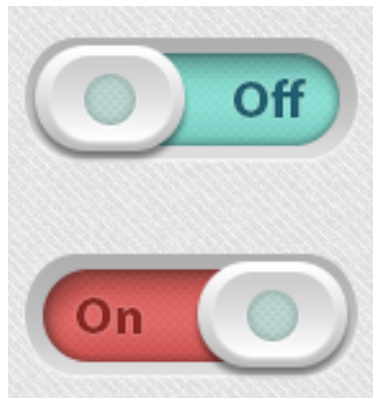
Emerging Alternative: Feature Toggles

```
#ifdef new_feature_on  
    /* code of new feature */  
#endif
```

pre-compilation

```
if(new_feature_on==true){  
    /* code of new feature */  
}
```

run-time



Toggle-based Development on Trunk



Toggle-based Development on Trunk



Toggle-based Development on Trunk

master branch —●————●————→



32



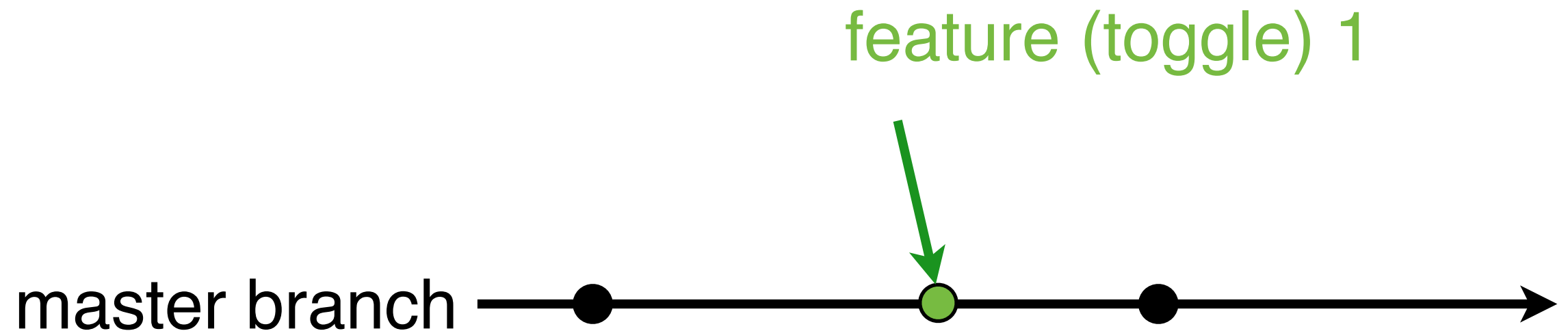
Toggle-based Development on Trunk

feature (toggle) 1

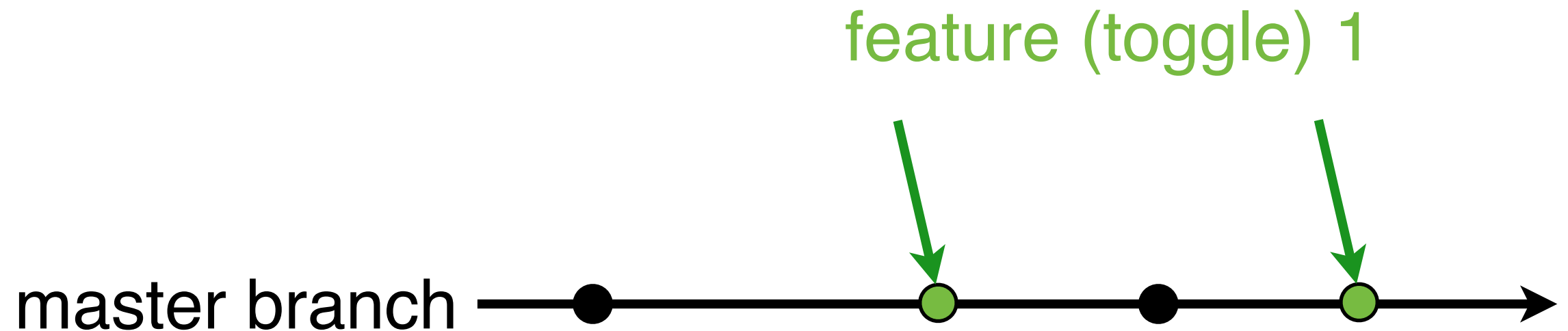
master branch —●————●————→



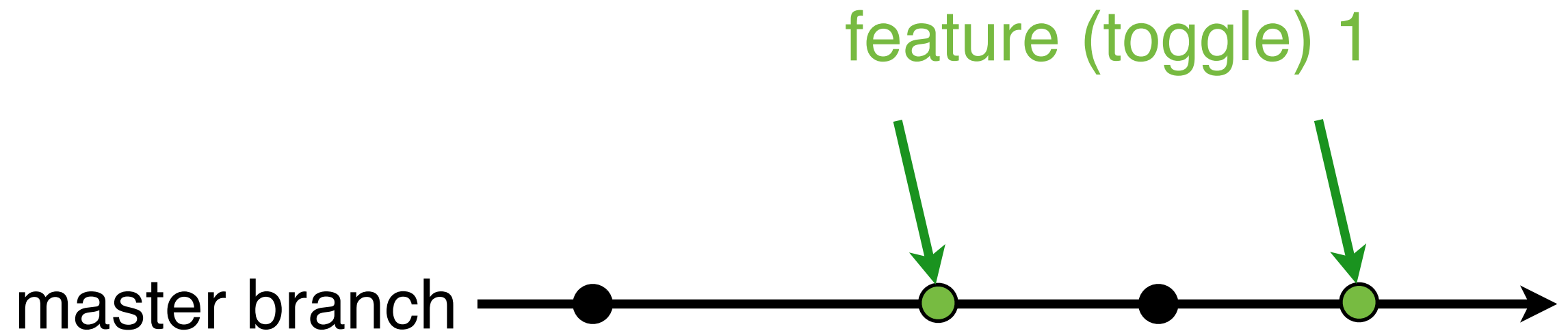
Toggle-based Development on Trunk



Toggle-based Development on Trunk



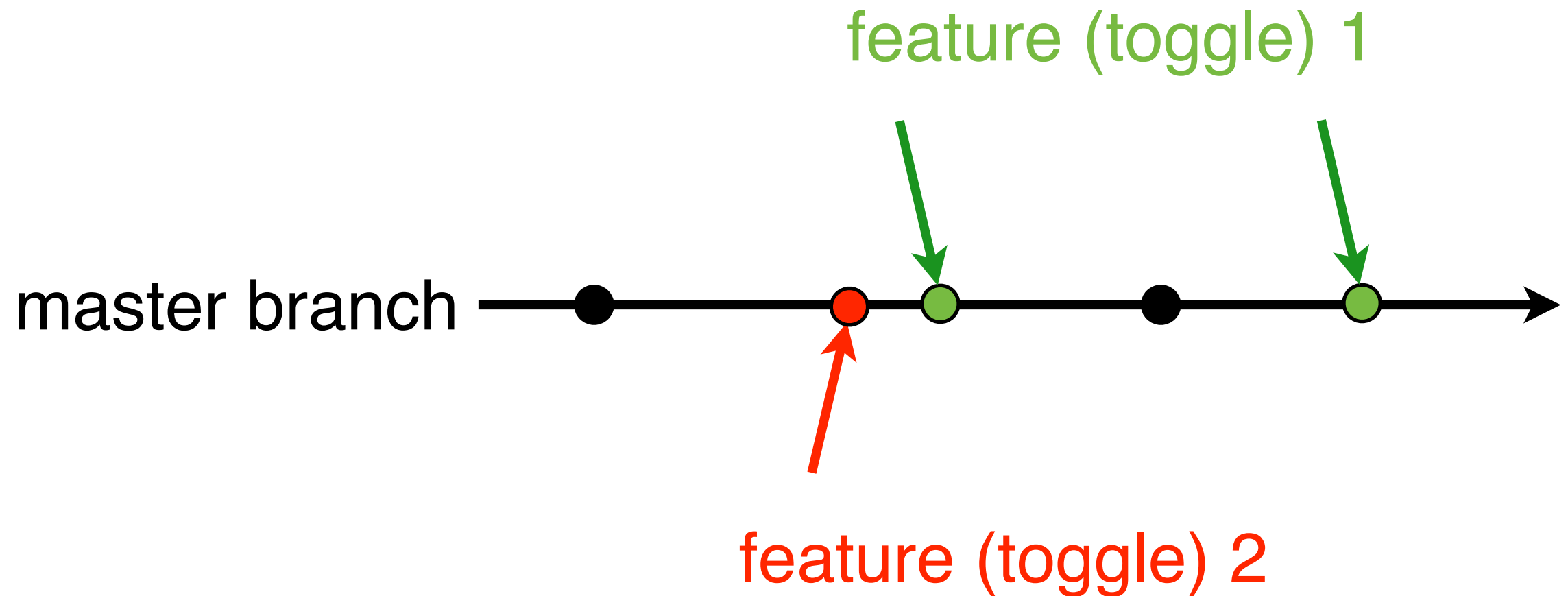
Toggle-based Development on Trunk



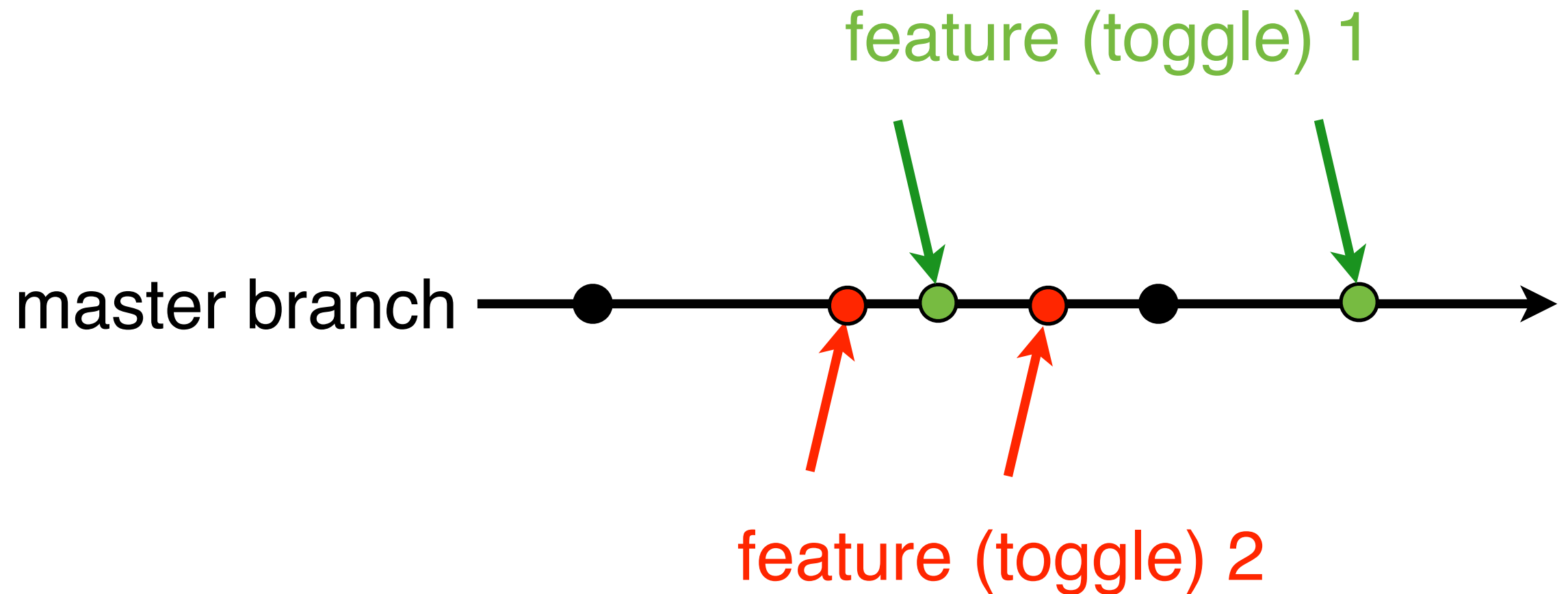
feature (toggle) 2



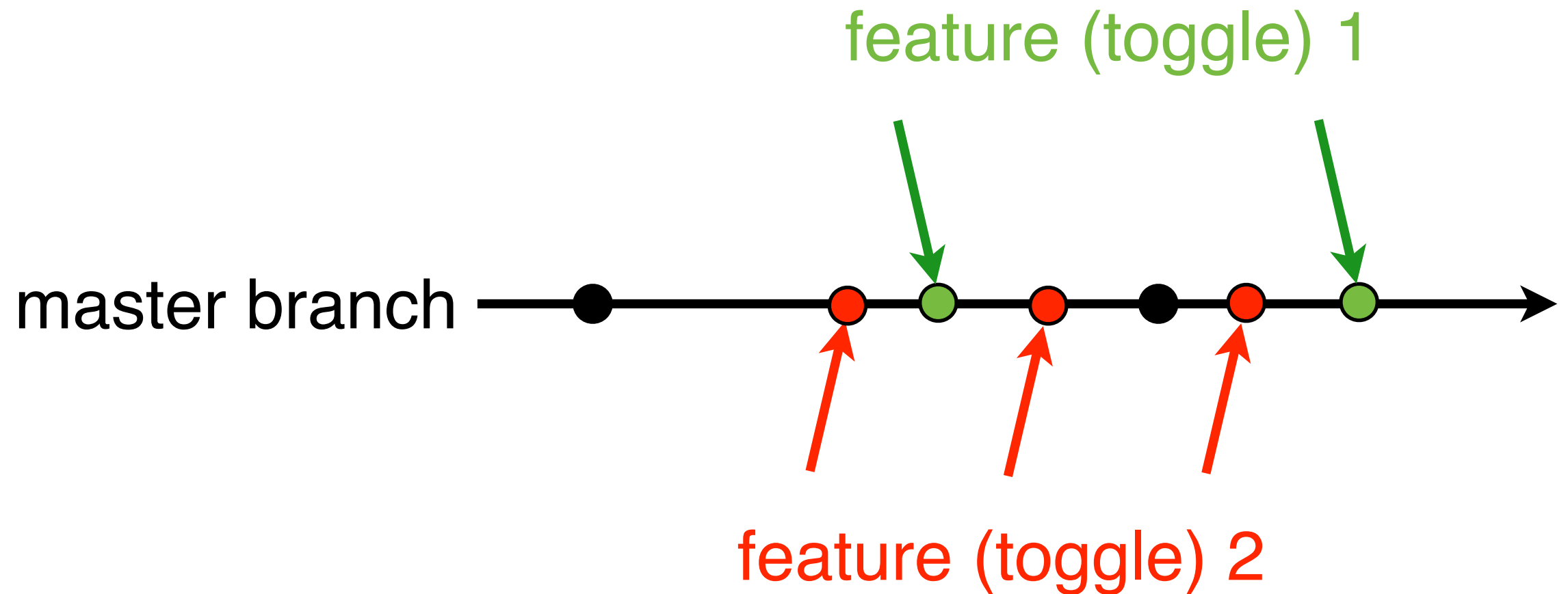
Toggle-based Development on Trunk



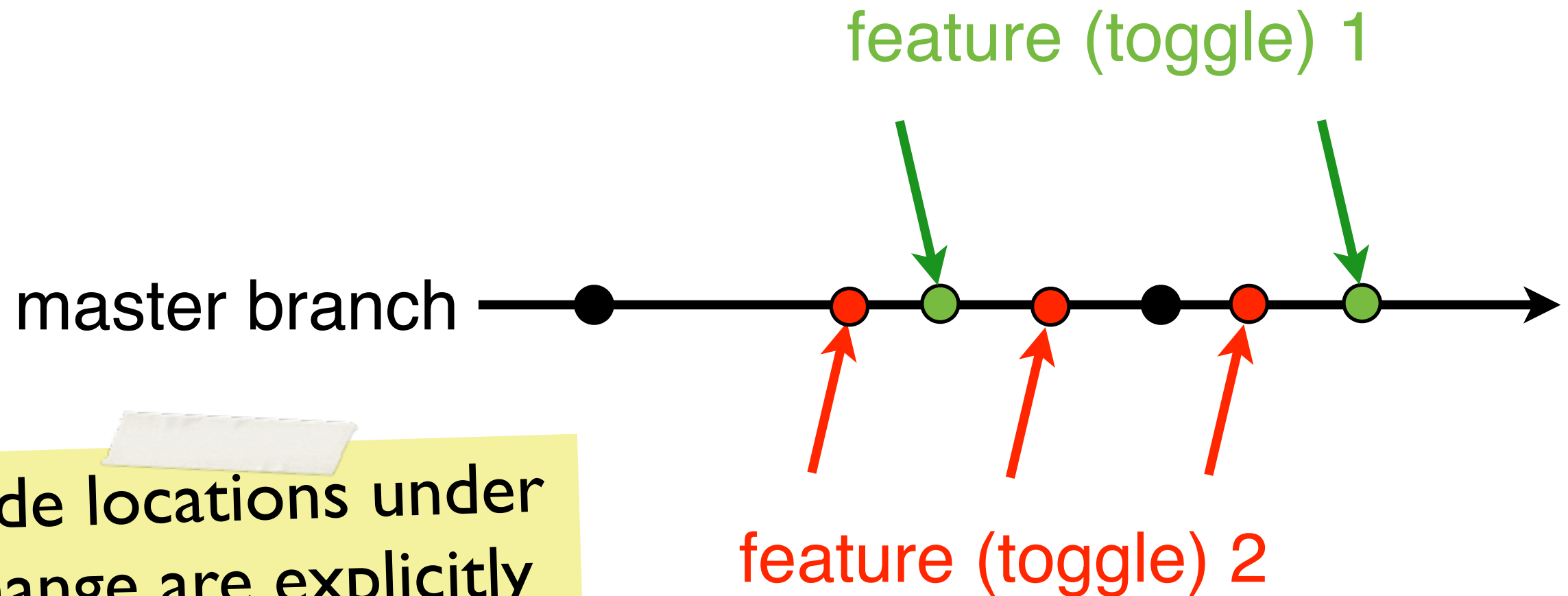
Toggle-based Development on Trunk



Toggle-based Development on Trunk



Toggle-based Development on Trunk



code locations under change are explicitly visible (toggled blocks)





integrating code changes



building/testing (CI)



deploying a new release



releasing to the user



integrating code changes



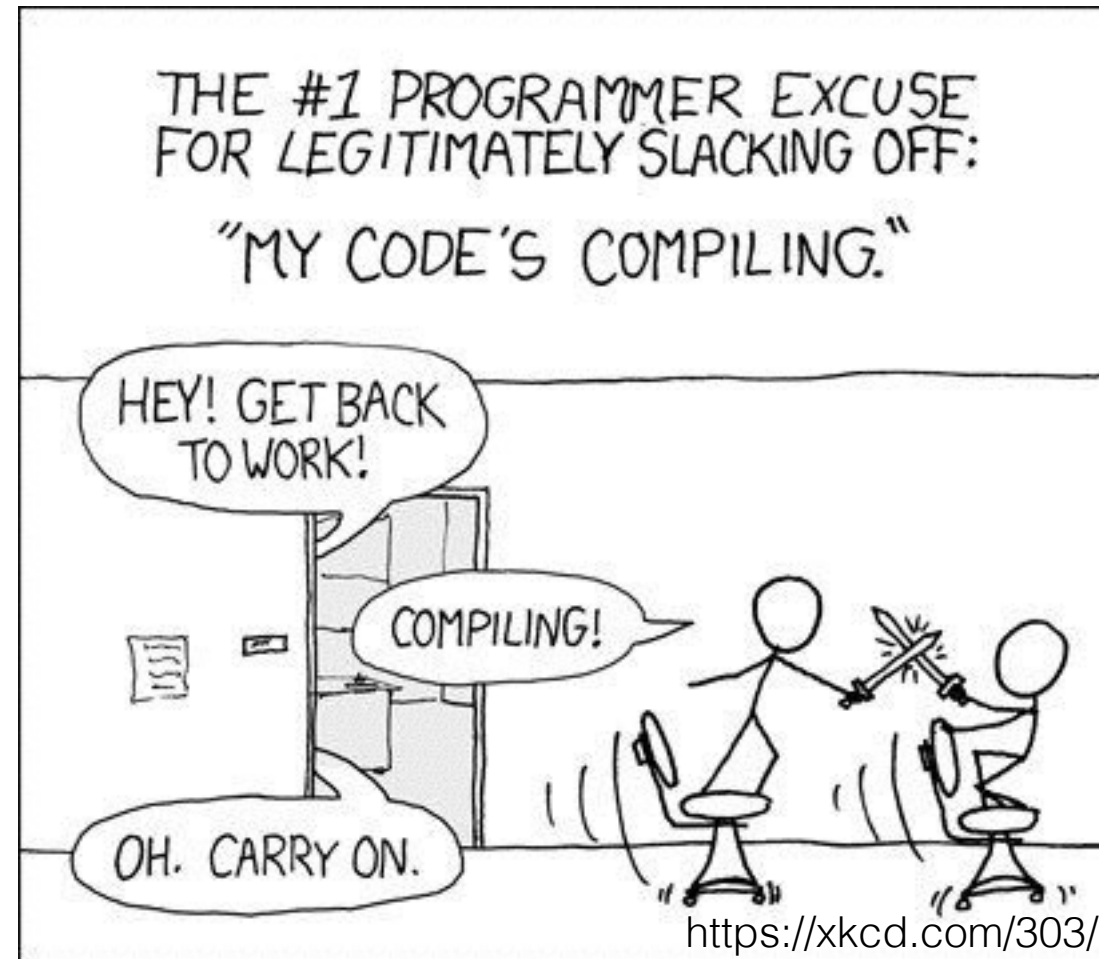
building/testing (CI)



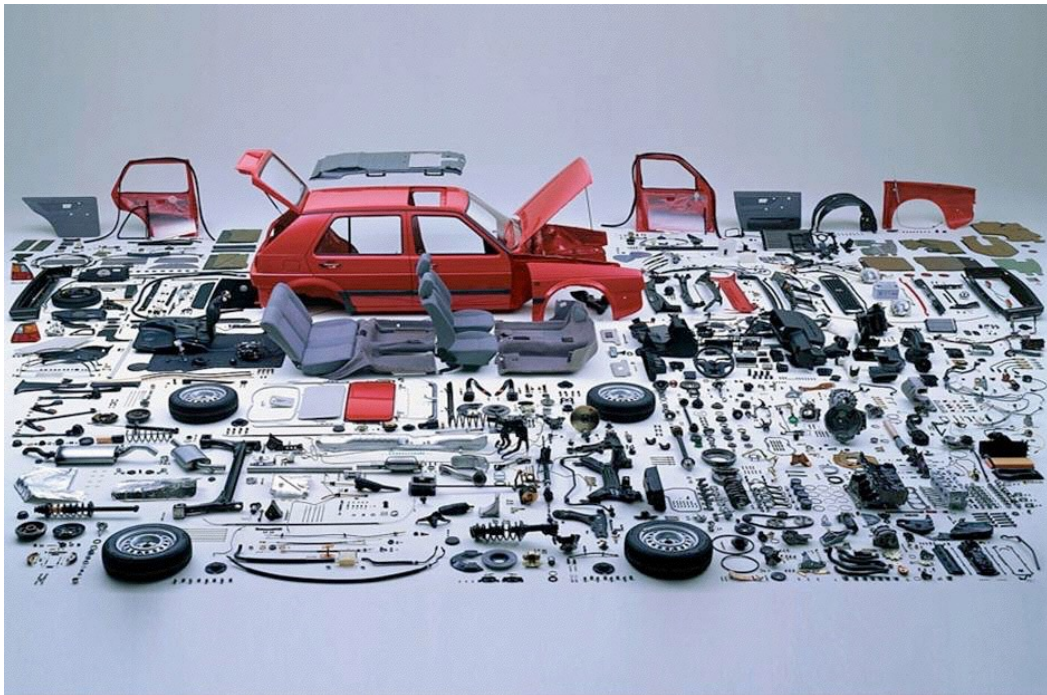
releasing to the user

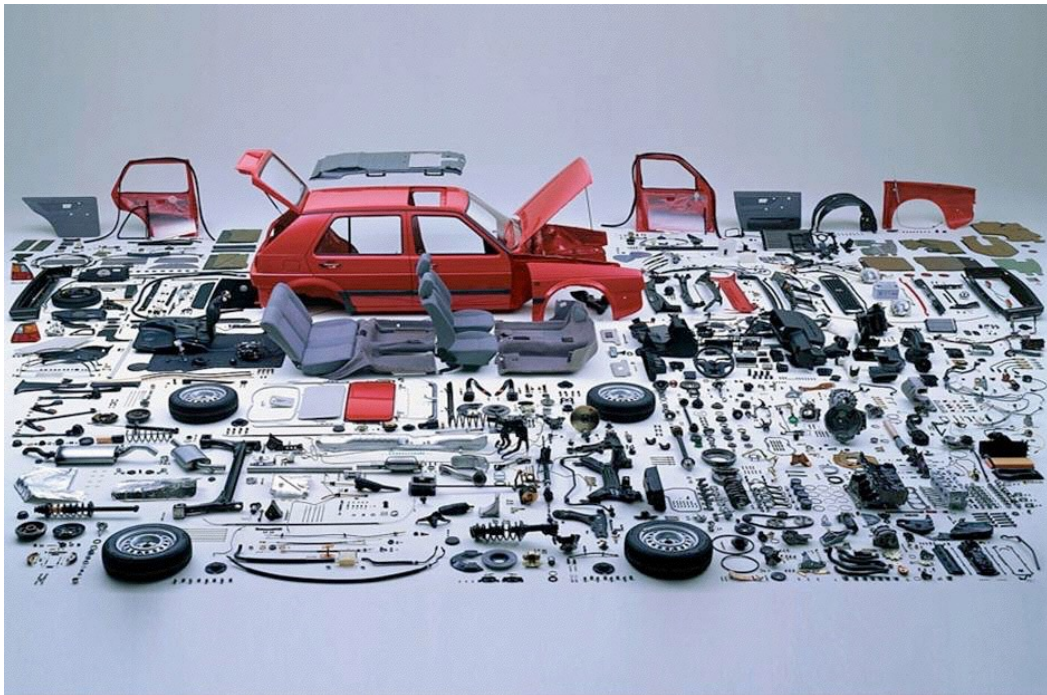


deploying a new release



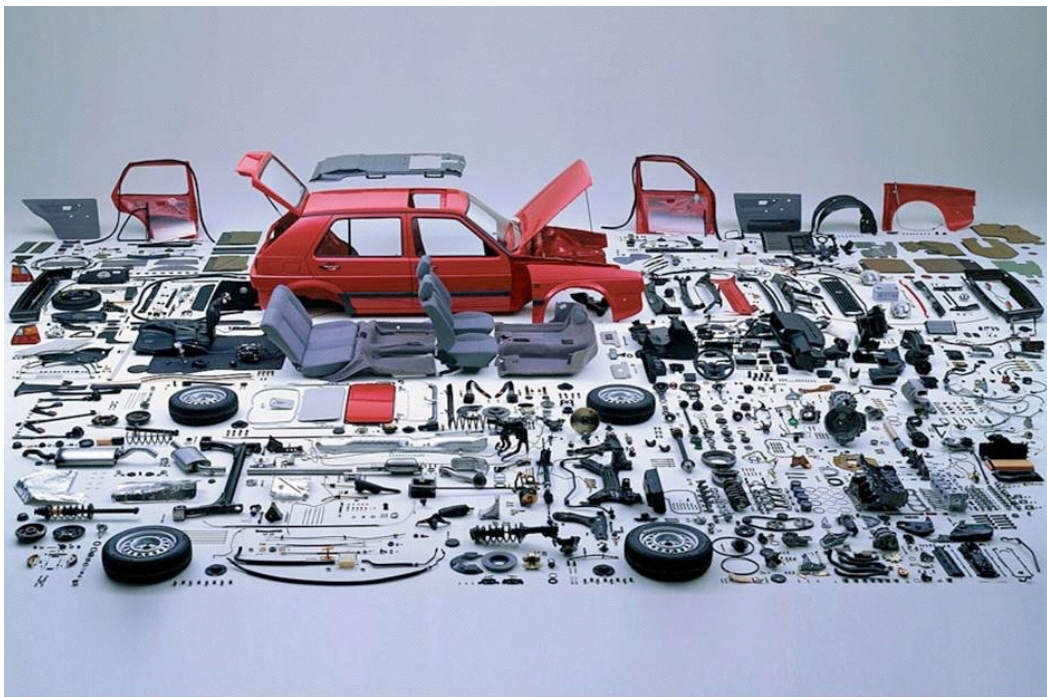








Build System Converts Source Code into Deliverables



Make



Autotools



maven



Step I - Configuration



Autotools





Autotools



Step 1 - Configuration

Features



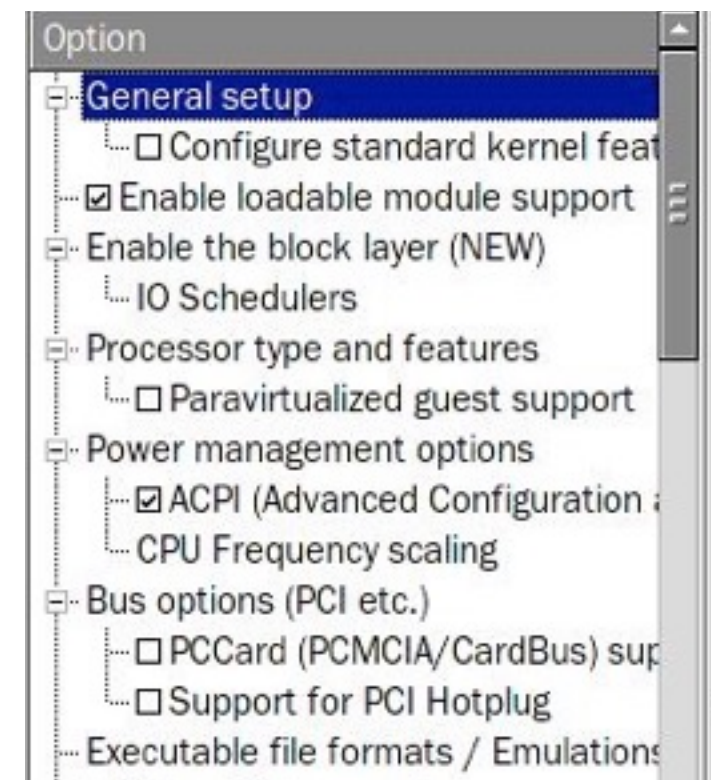


Autotools



Step I - Configuration

Features



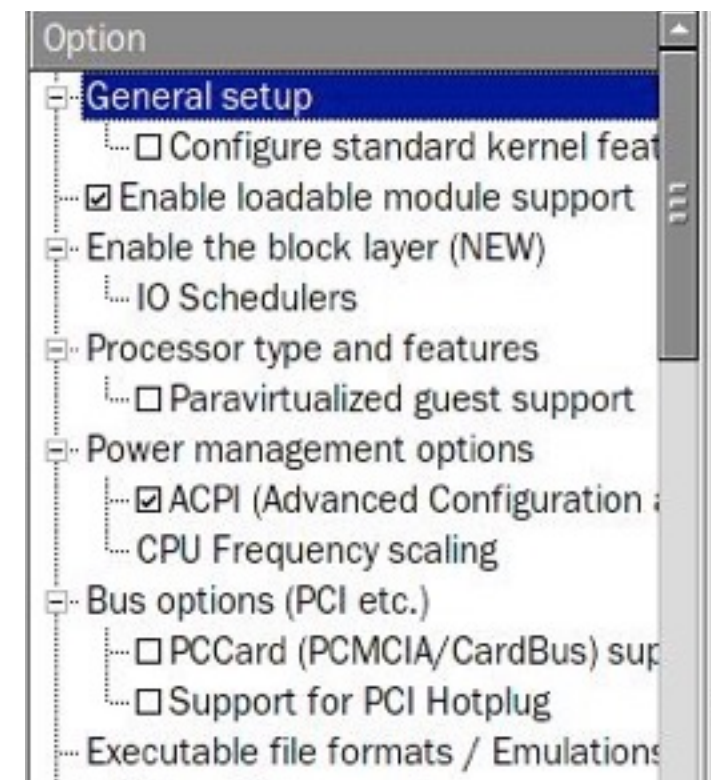
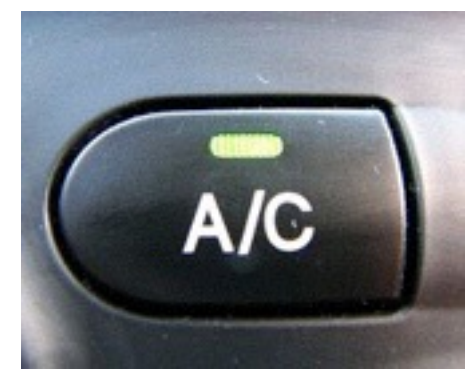


Autotools



Step I - Configuration

Features



Tools



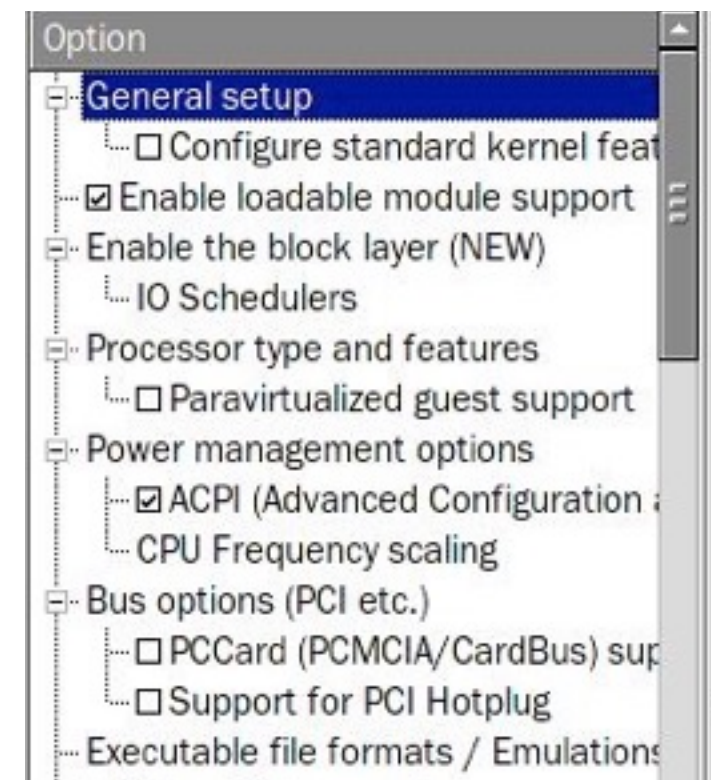


Autotools



Step I - Configuration

Features



Tools





```
menuconfig USB_SUPPORT
bool "USB support"
depends on HAS_IOMEM
default y
---help---
```

This option adds core support for Universal Serial Bus (USB).
You will also need drivers from the following menu to use it.

```
if USB_SUPPORT
```

```
config USB_COMMON
tristate
default y
depends on USB || USB_GADGET
```

```
# Host-side USB depends on having a host controller
```

```
config USB_ARCH_HAS_HCD
boolean
default y if USB_ARCH_HAS_OHCI
default y if USB_ARCH_HAS_EHCI
default y if USB_ARCH_HAS_XHCI
default y if PCMCIA && !M32R # sl811_cs
default y if ARM # SL-811
default y if BLACKFIN # SL-811
default y if SUPERH # r8a66597-hcd
default PCI
```

Example (Kconfig)





```
menuconfig USB_SUPPORT
bool "USB support"
depends on HAS_IOMEM
default y
---help---
```

This option adds core support for Universal Serial Bus (USB).
You will also need drivers from the following menu to use it.

```
if USB_SUPPORT
```

```
config USB_COMMON
```

configuration option

```
tristate
default y
depends on USB || USB_GADGET
```

```
# Host-side USB depends on having a host controller
```

```
config USB_ARCH_HAS_HCD
```

configuration option

```
boolean
default y if USB_ARCH_HAS_OHCI
default y if USB_ARCH_HAS_EHCI
default y if USB_ARCH_HAS_XHCI
default y if PCMCIA && !M32R # sl811_cs
default y if ARM # SL-811
default y if BLACKFIN # SL-811
default y if SUPERH # r8a66597-hcd
default PCI
```





Example (Kconfig)

```
menuconfig USB_SUPPORT
bool "USB support"
depends on HAS_IOMEM
default y
---help---
```

This option adds core support for Universal Serial Bus (USB).
You will also need drivers from the following menu to use it.

```
if USB_SUPPORT
```

```
config USB_COMMON configuration option
tristate module/built-in/no
default y
depends on USB || USB_GADGET
```

```
# Host-side USB depends on having a host controller
```

```
config USB_ARCH_HAS_HCD configuration option
boolean yes/no
default y if USB_ARCH_HAS_OHCI
default y if USB_ARCH_HAS_EHCI
default y if USB_ARCH_HAS_XHCI
default y if PCMCIA && !M32R # sl811_cs
default y if ARM # SL-811
default y if BLACKFIN # SL-811
default y if SUPERH # r8a66597-hcd
default PCI
```





```
menuconfig USB_SUPPORT
bool "USB support"
depends on HAS_IOMEM
default y
---help---
```

This option adds core support for Universal Serial Bus (USB).
You will also need drivers from the following menu to use it.

```
if USB_SUPPORT
```

```
config USB_COMMON configuration option
tristate module/built-in/no
default y
depends on USB || USB_GADGET constraint
```

```
# Host-side USB depends on having a host controller
```

```
config USB_ARCH_HAS_HCD configuration option
boolean yes/no
```

```
default y if USB_ARCH_HAS_OHCI
default y if USB_ARCH_HAS_EHCI
default y if USB_ARCH_HAS_XHCI
default y if PCMCIA && !M32R # sl811_cs
default y if ARM # SL-811
default y if BLACKFIN # SL-811
default y if SUPERH # r8a66597-hcd
default PCI
```

constraint



Example (Kconfig)



Step 2 - Construction



Step 2 - Construction

Recipes



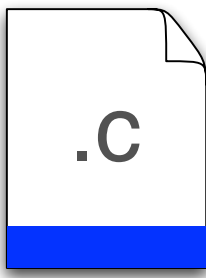
=





Step 2 - Construction

Recipes



=



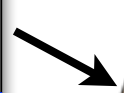
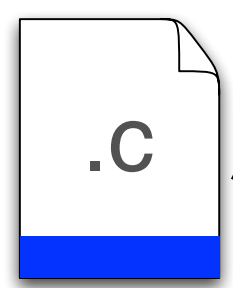


Step 2 - Construction

Recipes



=



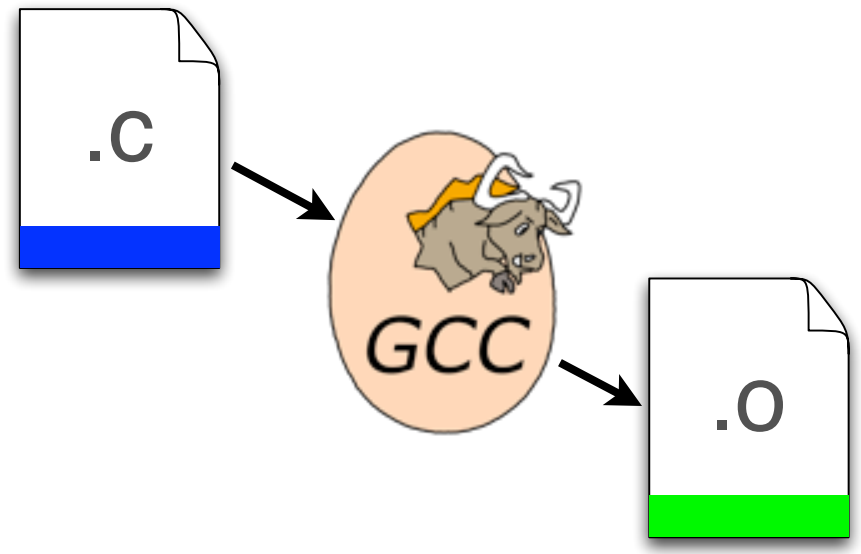


Step 2 - Construction

Recipes



=





Step 2 - Construction

Recipes



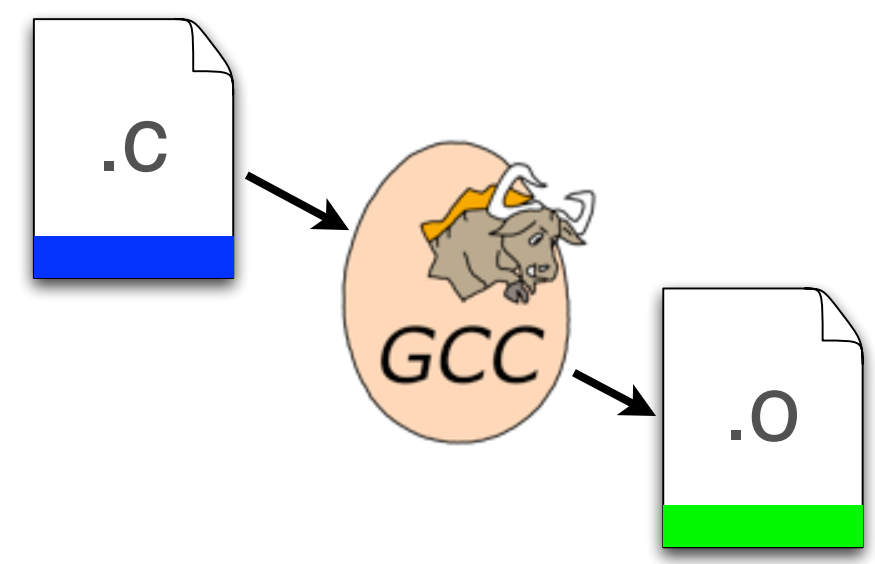
=



Dependencies



← - - - -





Step 2 - Construction

Recipes



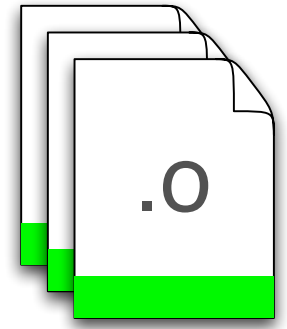
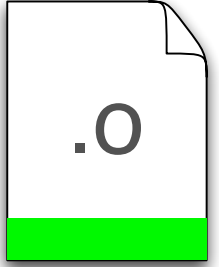
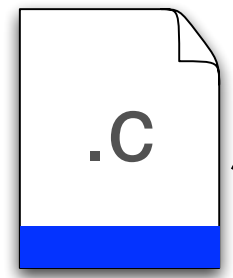
=



Dependencies



← - - - -





Step 2 - Construction

Recipes



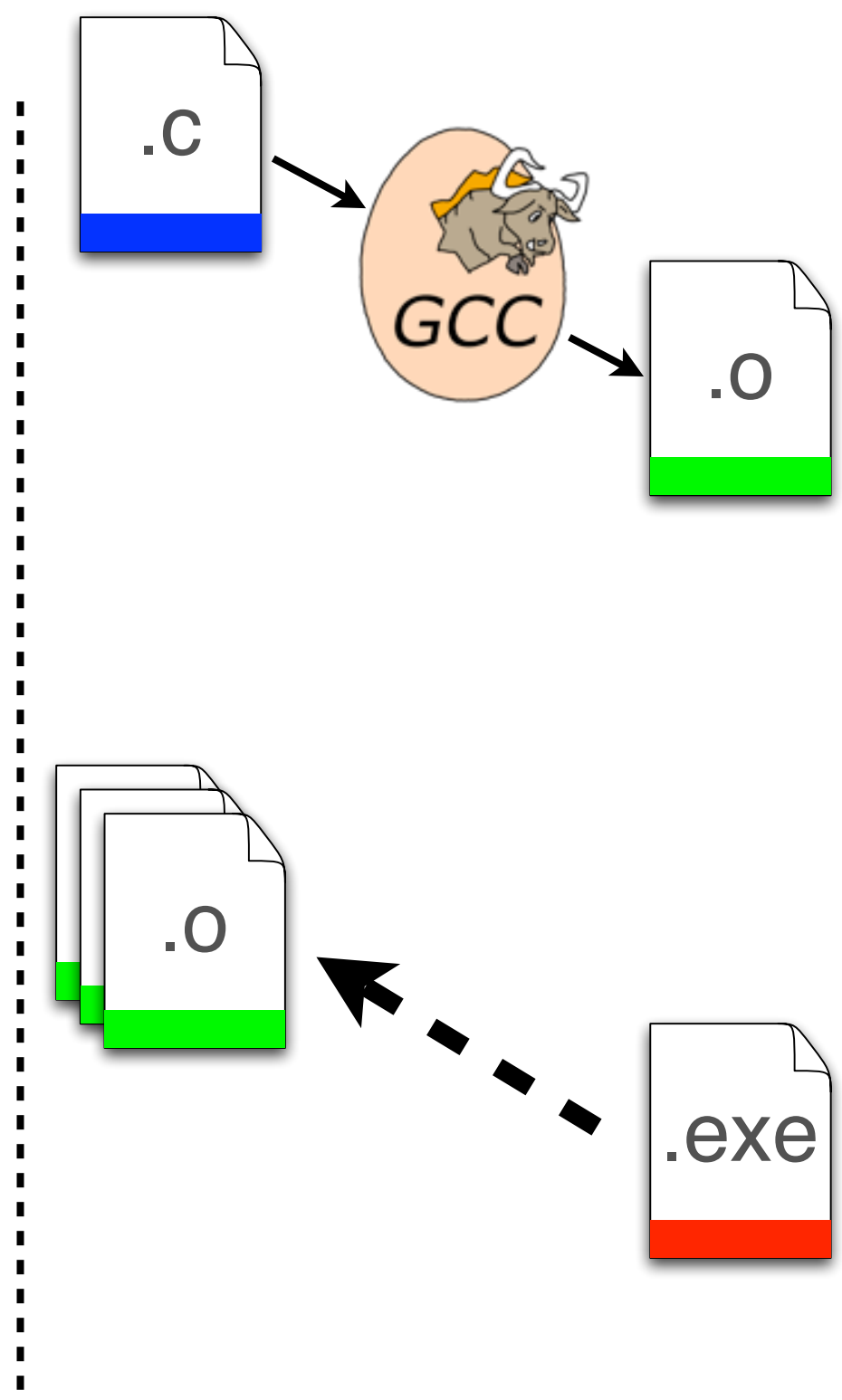
=



Dependencies



← - - - -





Example (GNU Make)

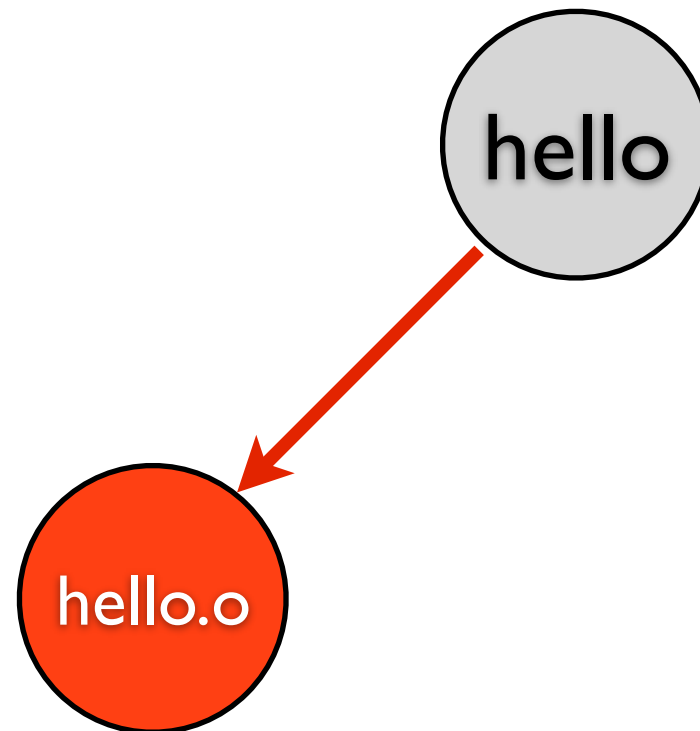


Example (GNU Make)

hello

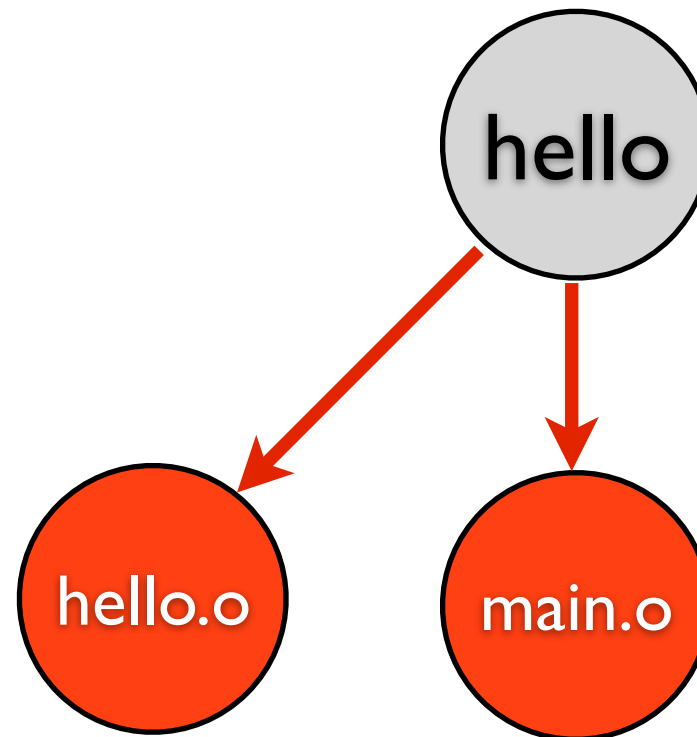


Example (GNU Make)



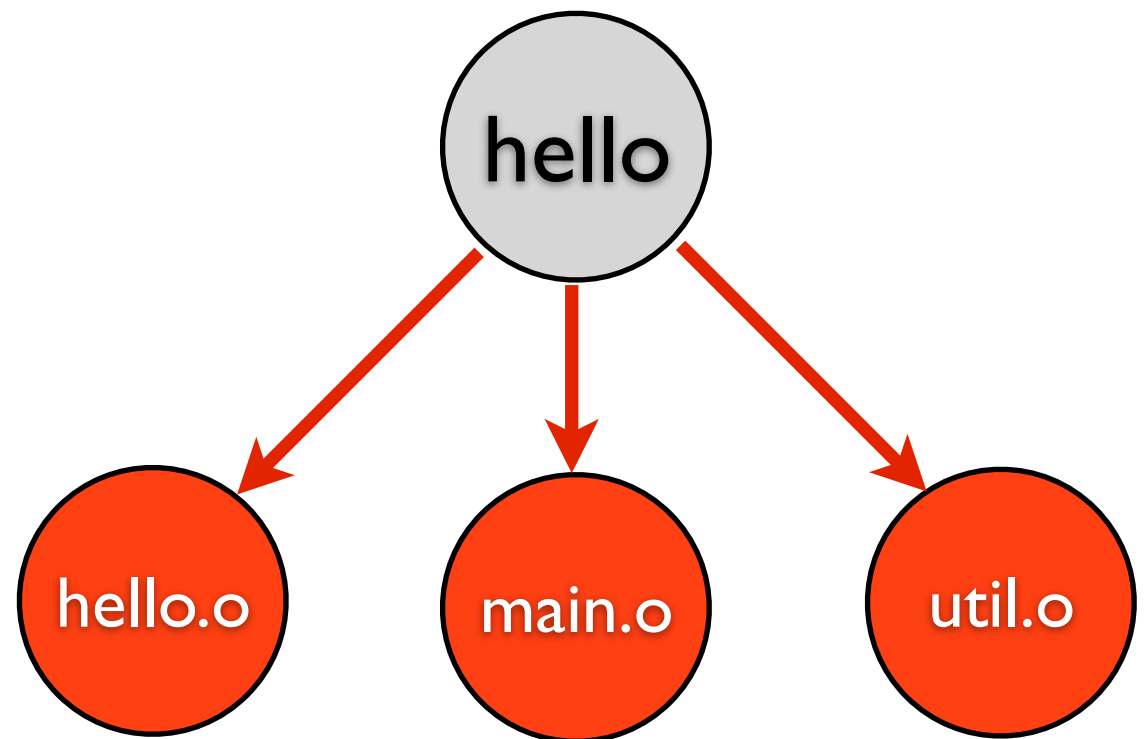


Example (GNU Make)



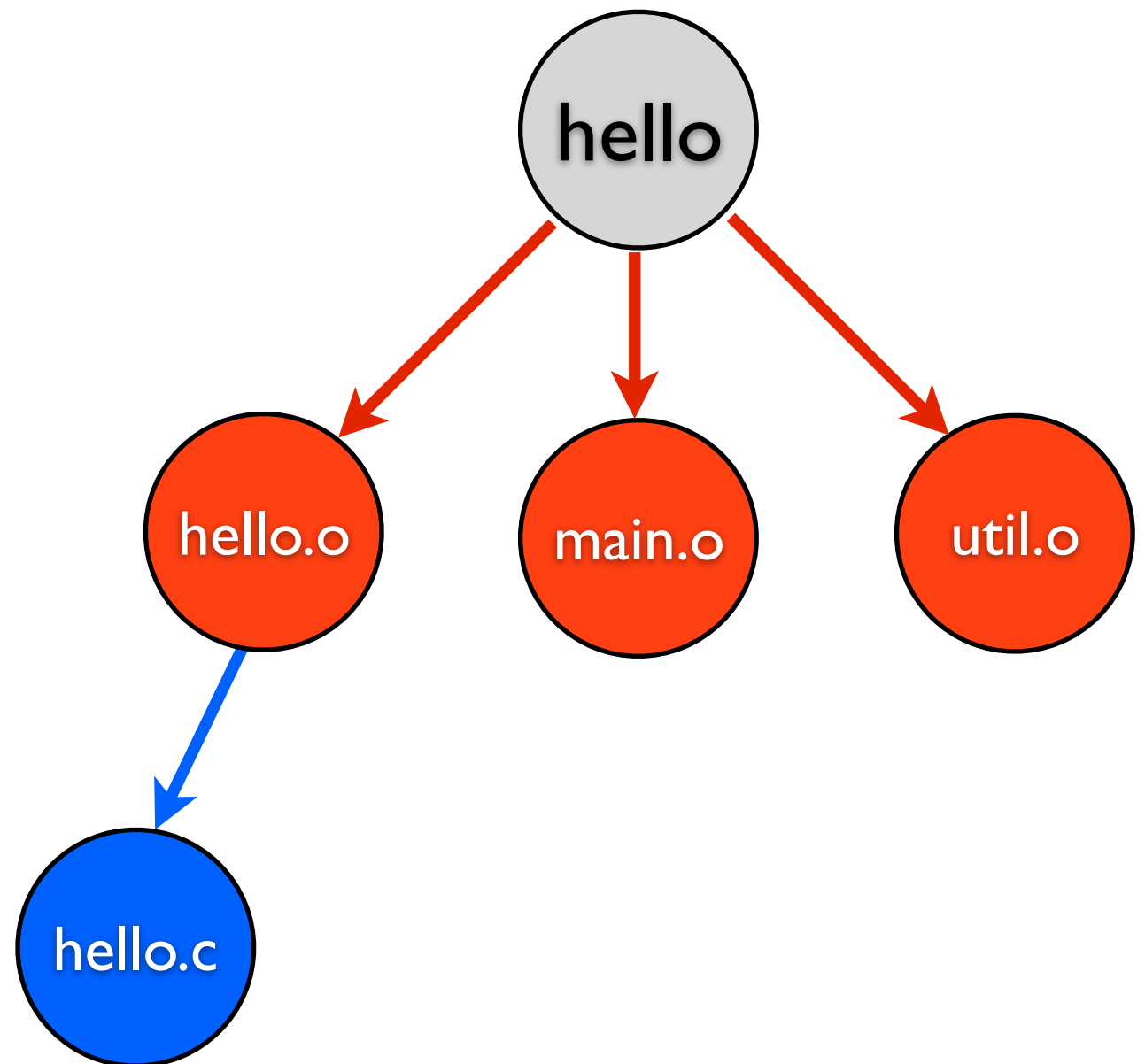


Example (GNU Make)



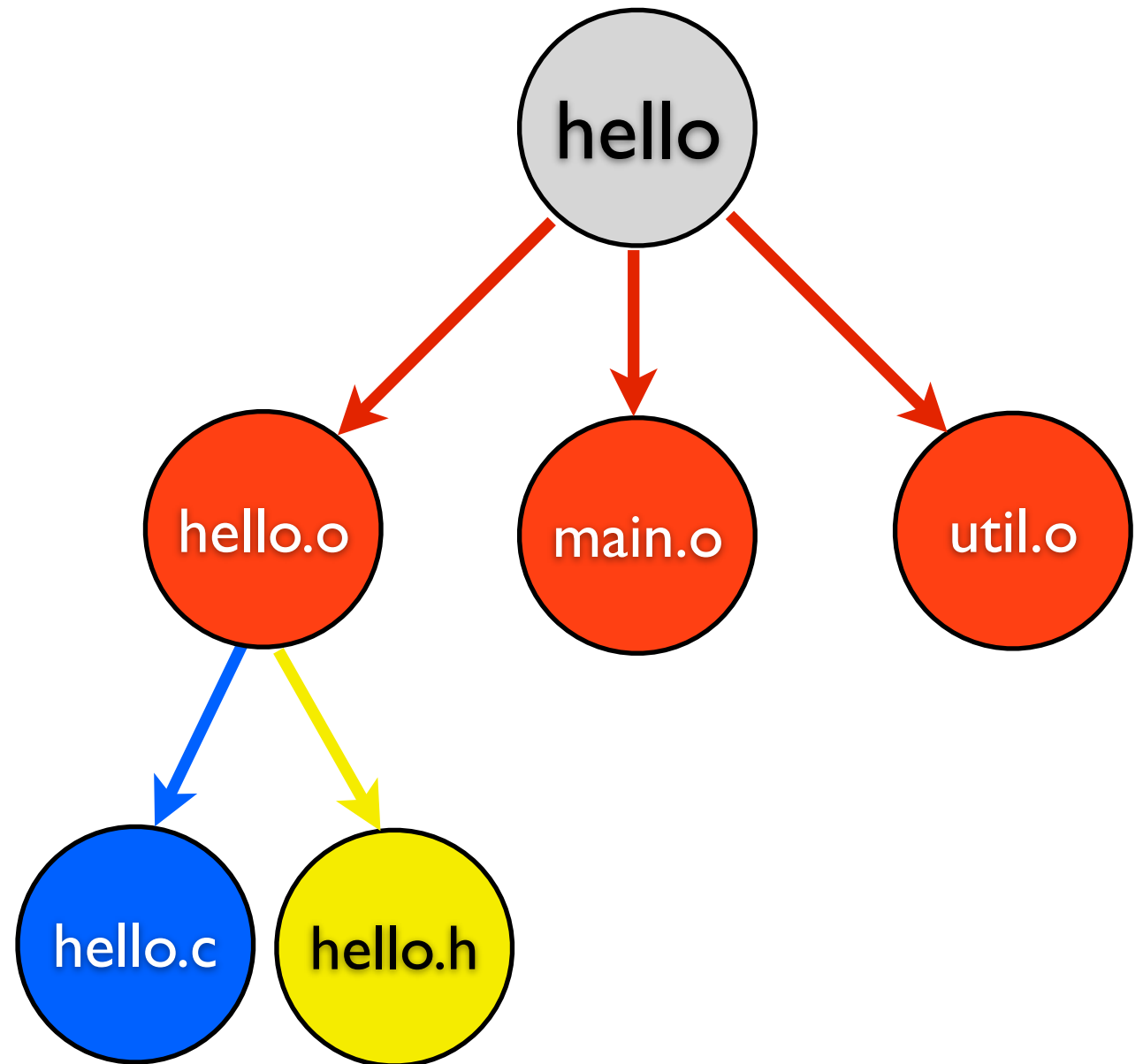


Example (GNU Make)



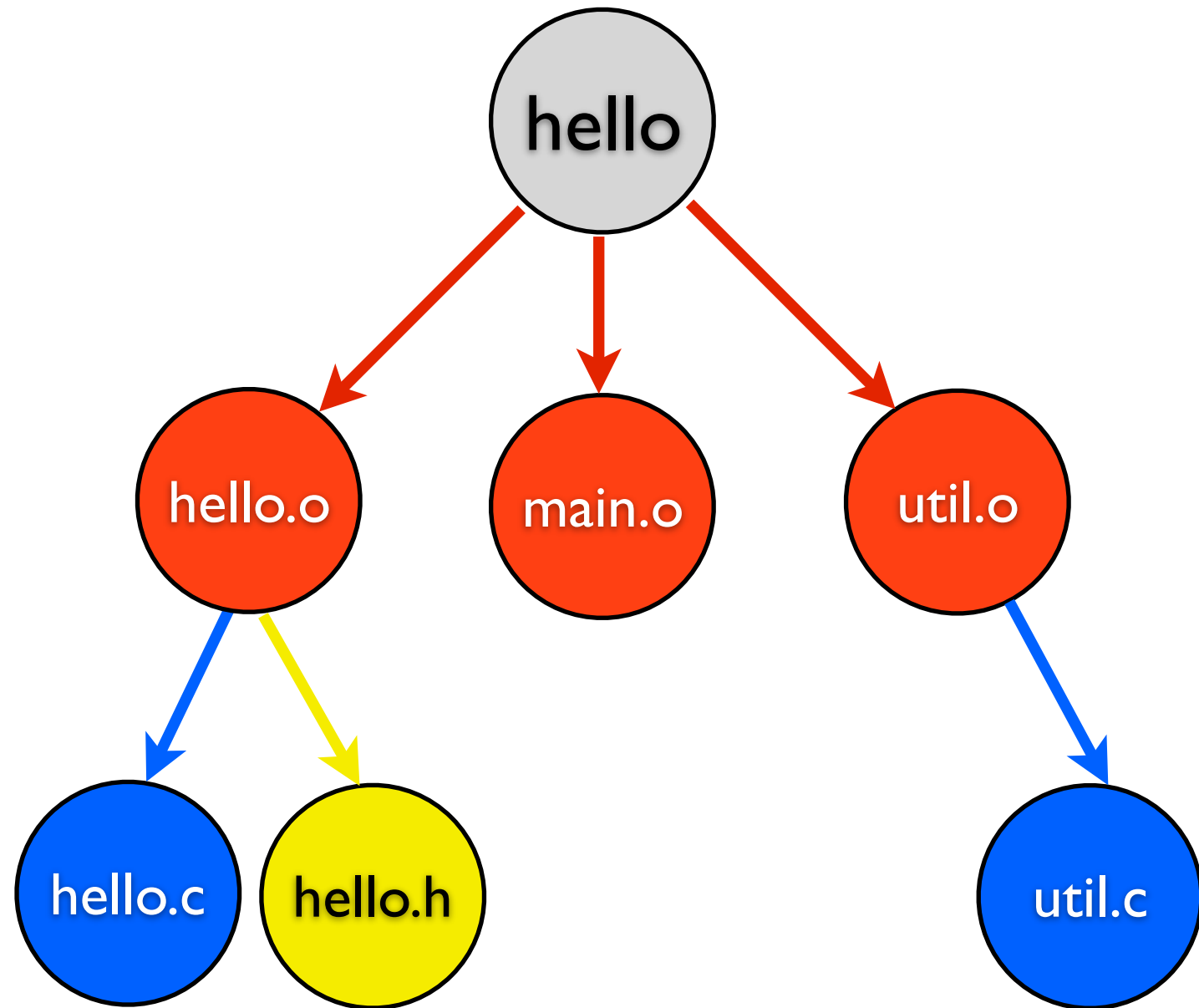


Example (GNU Make)



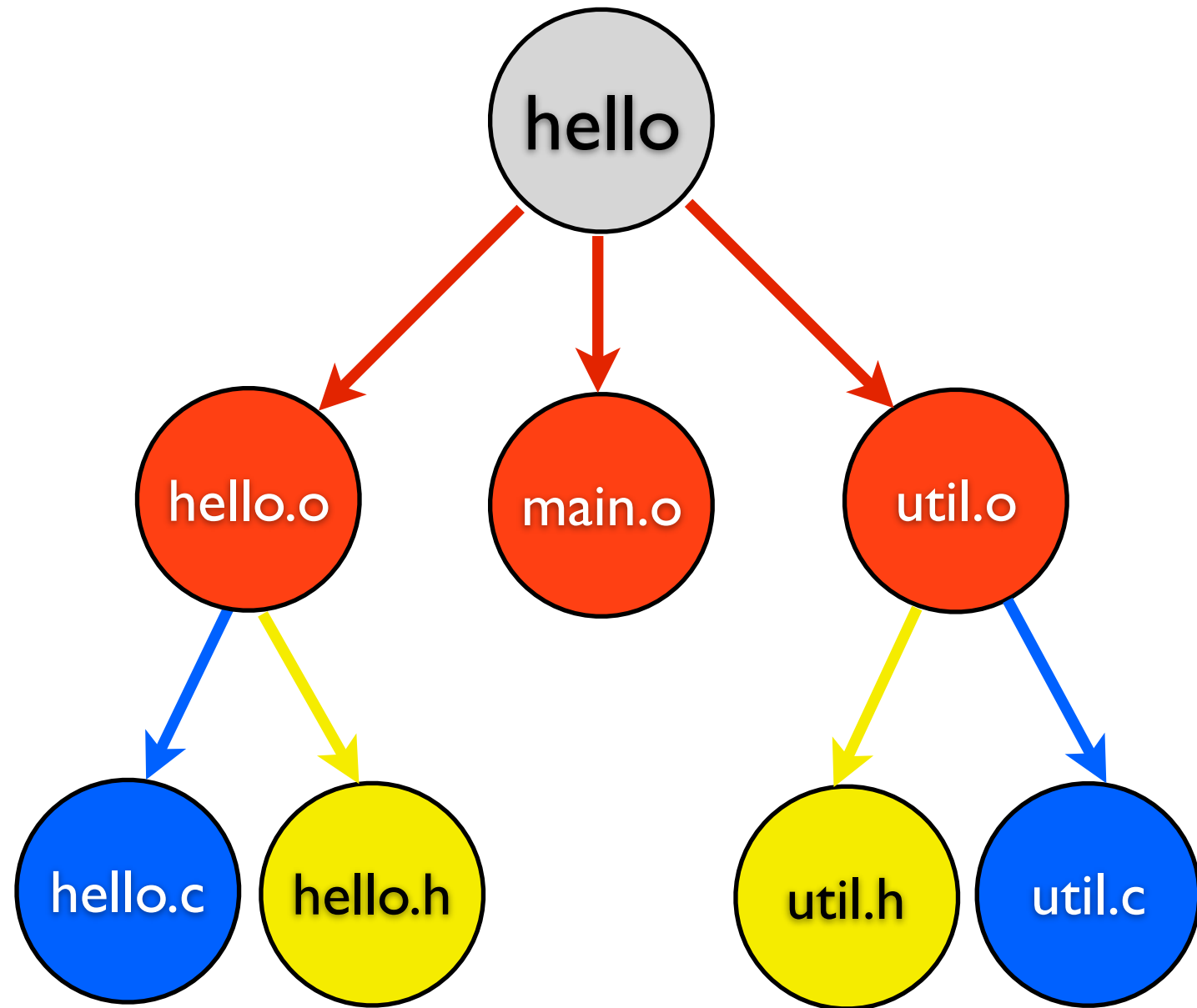


Example (GNU Make)



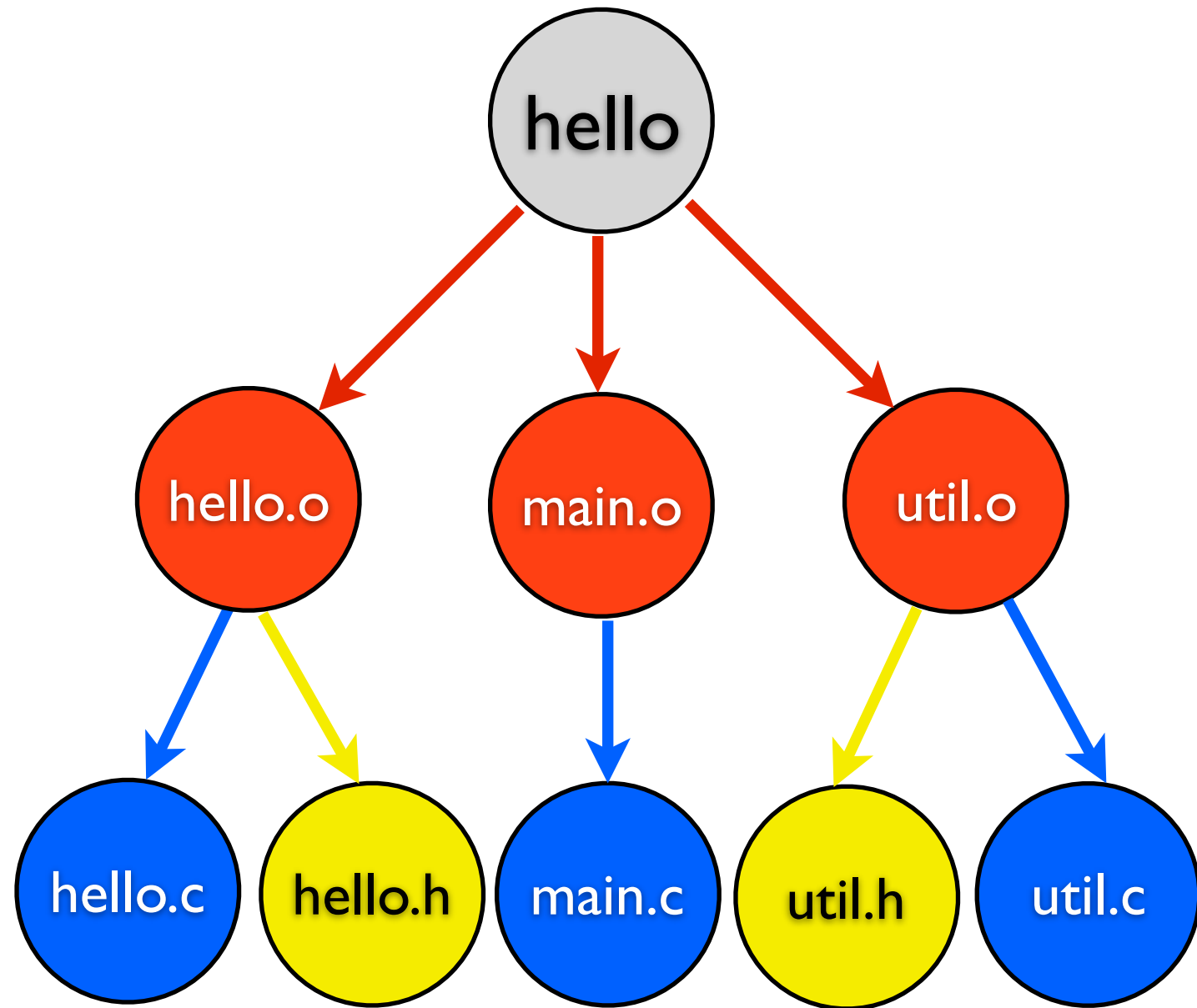


Example (GNU Make)



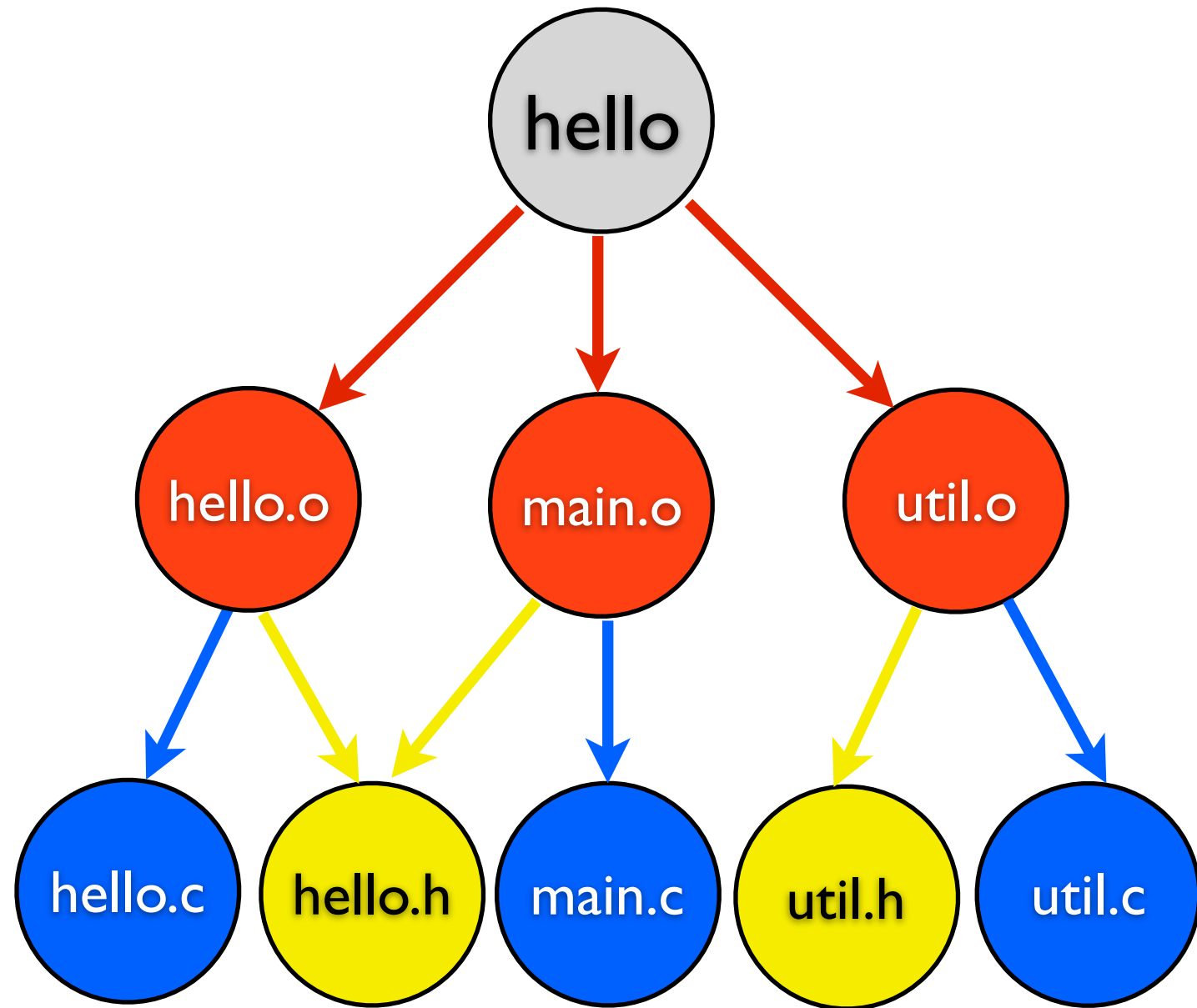


Example (GNU Make)



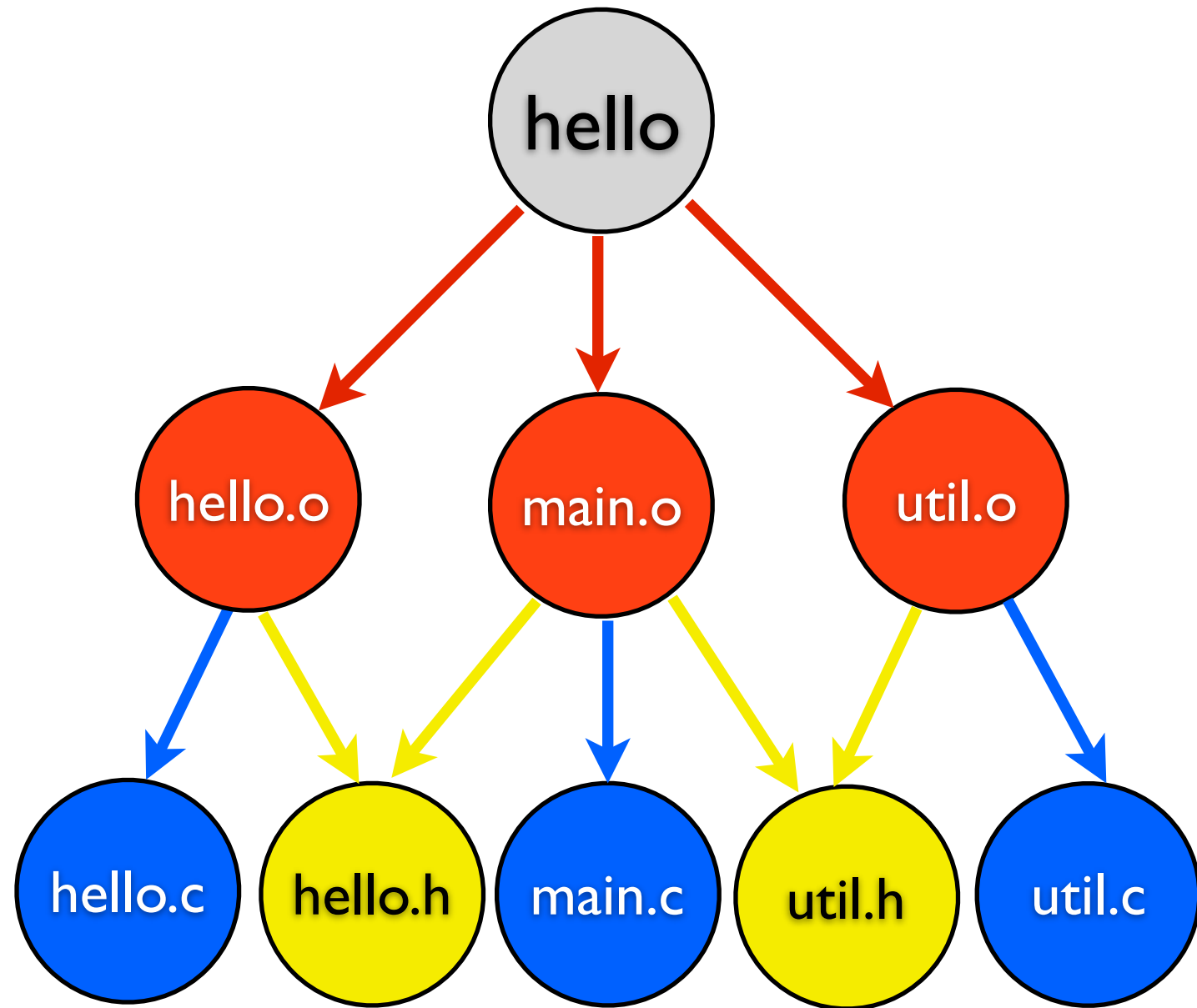


Example (GNU Make)





Example (GNU Make)





Example (GNU Make)

hello: hello.o util.o main.o

gcc -o hello hello.o util.o main.o

hello.o: hello.c hello.h

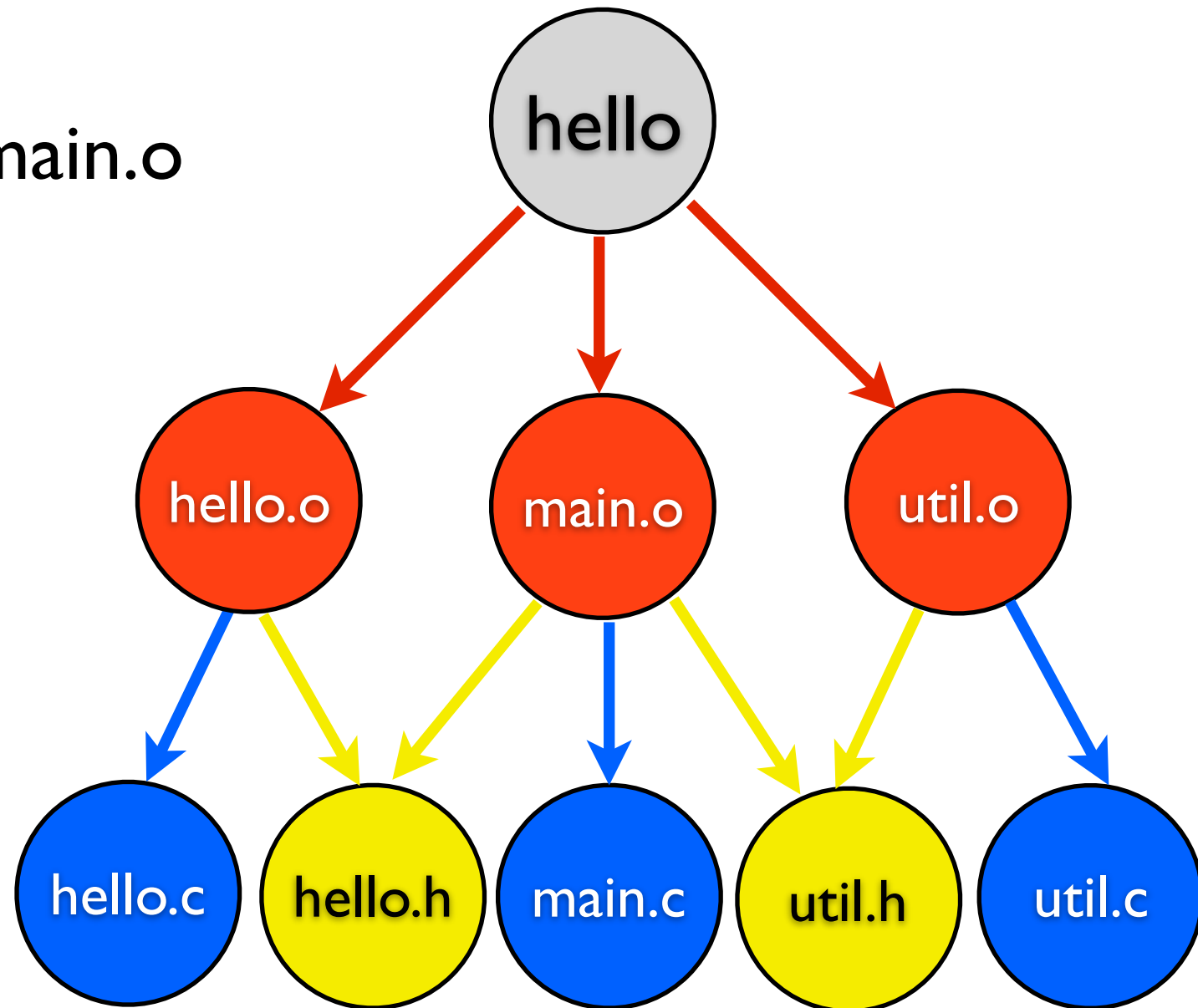
gcc -o hello.o -c hello.c

util.o: util.c util.h

gcc -o util.o -c util.c

main.o: main.c hello.h util.h

gcc -o main.o -c main.c





Example (GNU Make)

hello: hello.o util.o main.o

gcc -o hello hello.o util.o main.o

rule

hello.o: hello.c hello.h

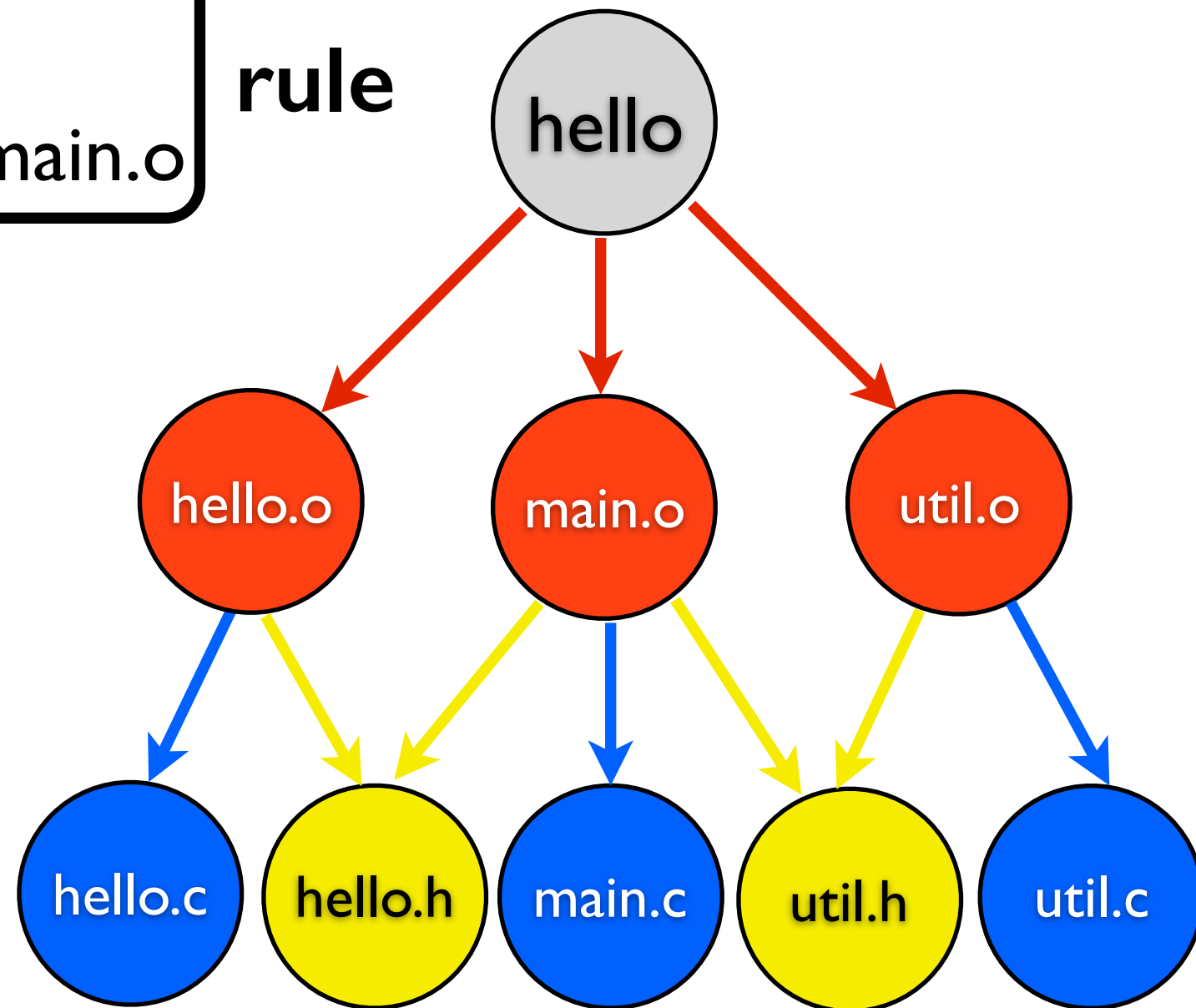
gcc -o hello.o -c hello.c

util.o: util.c util.h

gcc -o util.o -c util.c

main.o: main.c hello.h util.h

gcc -o main.o -c main.c





Example (GNU Make)

hello: hello.o util.o main.o

gcc -o hello hello.o util.o main.o

rule

hello.o: hello.c hello.h

gcc -o hello.o -c hello.c

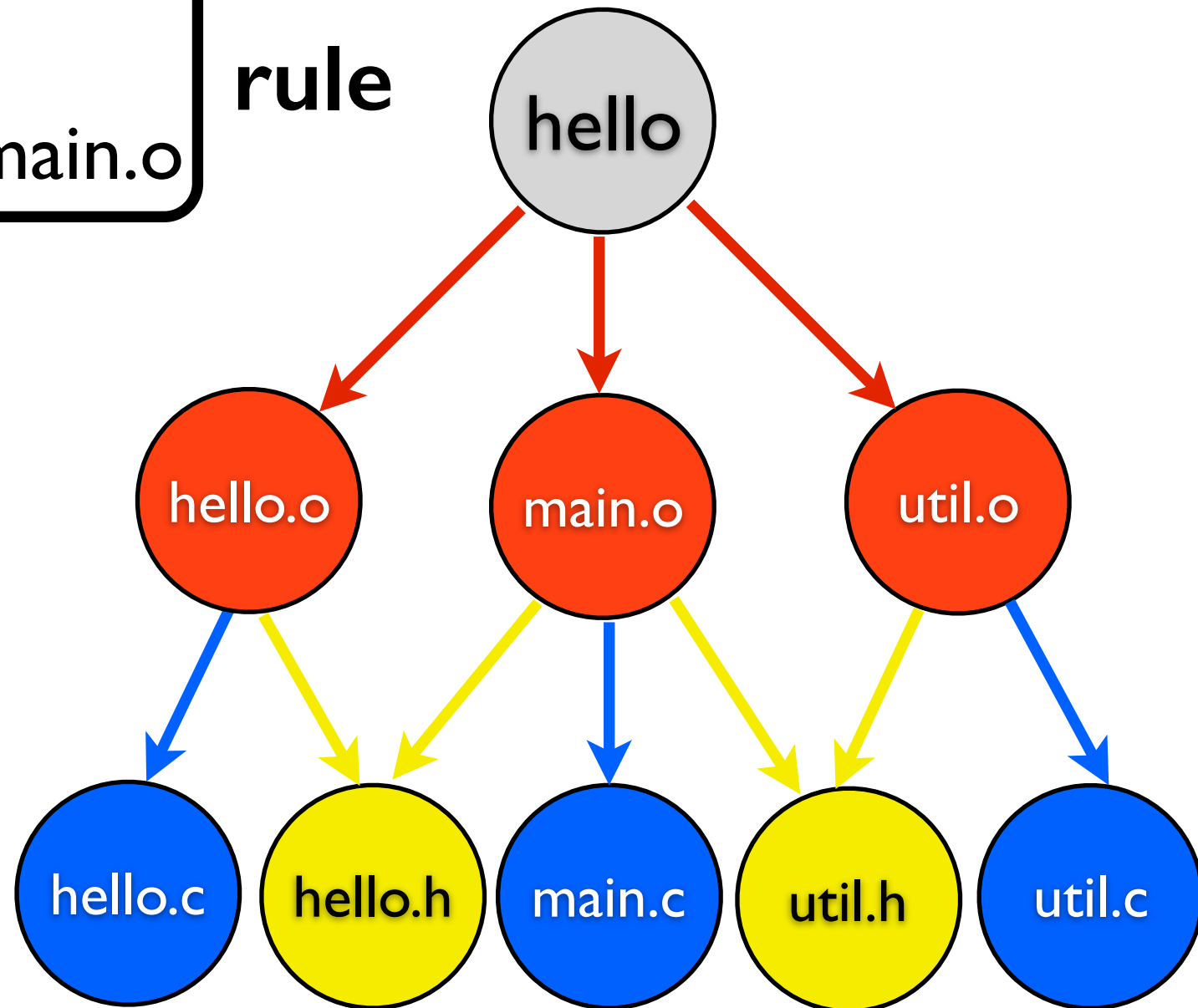
util.o: util.c util.h

gcc -o util.o -c util.c

target

main.o: main.c hello.h util.h

gcc -o main.o -c main.c





Example (GNU Make)

hello: hello.o util.o main.o

gcc -o hello hello.o util.o main.o

rule

hello.o: hello.c hello.h

gcc -o hello.o -c hello.c

util.o: util.c util.h

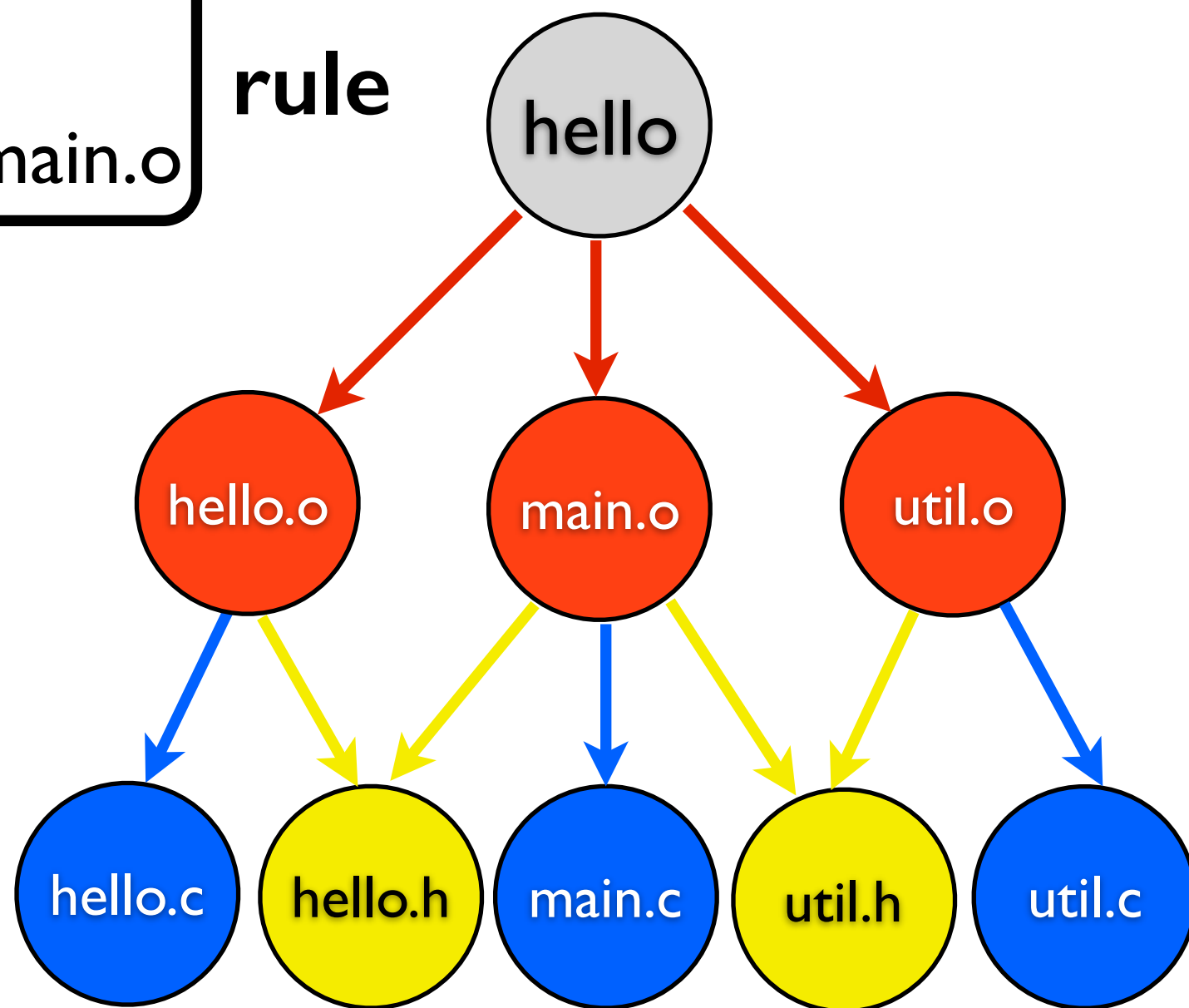
gcc -o util.o -c util.c

target

dependencies

main.o: main.c hello.h util.h

gcc -o main.o -c main.c





Example (GNU Make)

hello: hello.o util.o main.o

gcc -o hello hello.o util.o main.o

rule

hello.o: hello.c hello.h

gcc -o hello.o -c hello.c

util.o: util.c util.h

gcc -o util.o -c util.c

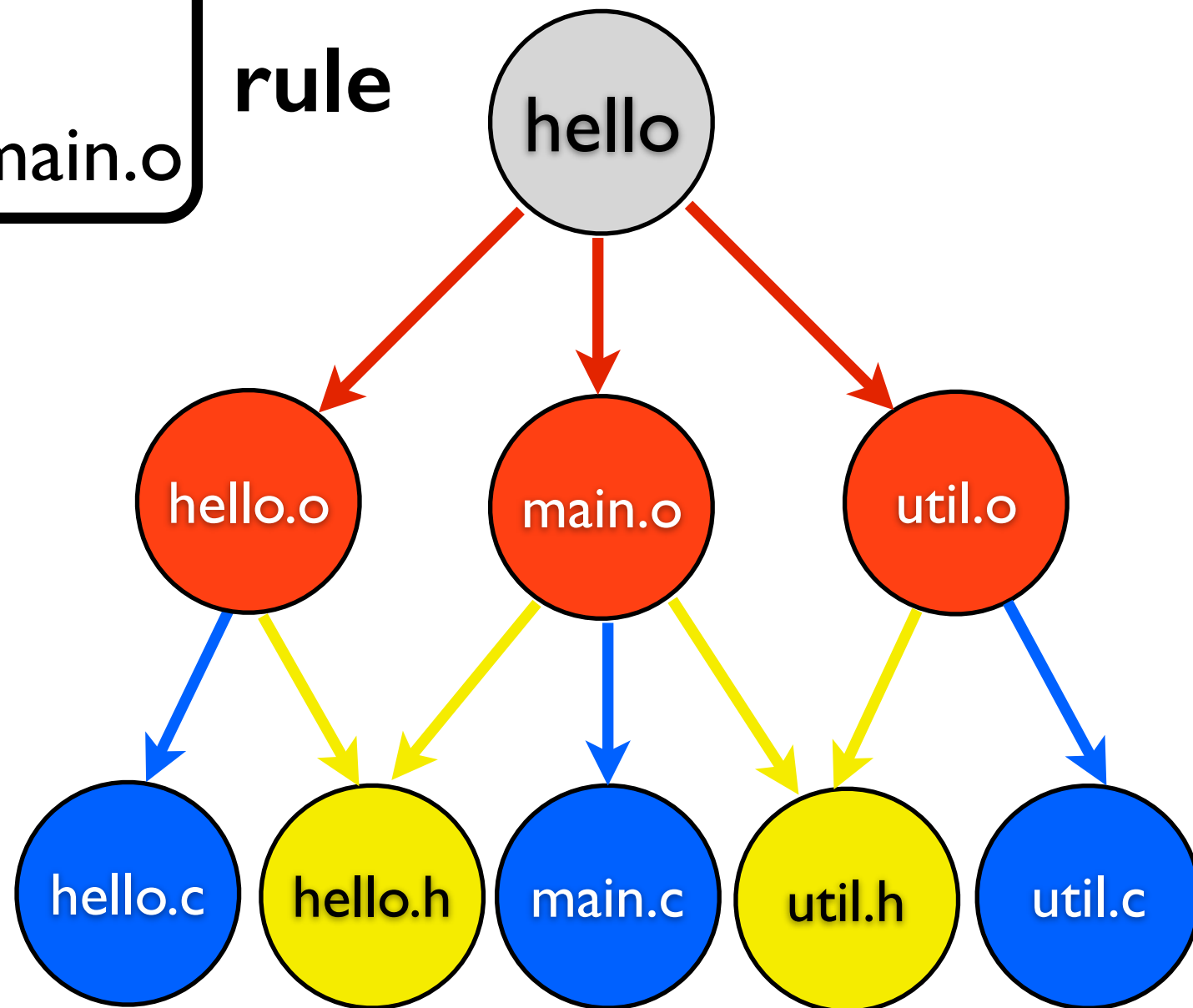
target

dependencies

main.o: main.c hello.h util.h

gcc -o main.o -c main.c

build recipe





A Full Build

hello: hello.o util.o main.o

gcc -o hello hello.o util.o main.o

hello.o: hello.c hello.h

gcc -o hello.o -c hello.c

util.o: util.c util.h

gcc -o util.o -c util.c

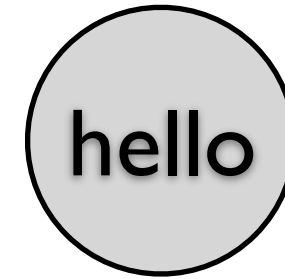
main.o: main.c hello.h util.h

gcc -o main.o -c main.c



A Full Build

hello:hello.o util.o main.o



gcc -o hello hello.o util.o main.o

hello.o: hello.c hello.h

gcc -o hello.o -c hello.c

util.o: util.c util.h

gcc -o util.o -c util.c

main.o: main.c hello.h util.h

gcc -o main.o -c main.c



A Full Build

hello: **hello.o** util.o main.o

gcc -o hello hello.o util.o main.o

hello.o: hello.c hello.h

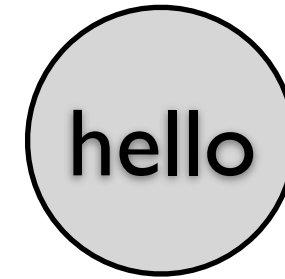
gcc -o hello.o -c hello.c

util.o: util.c util.h

gcc -o util.o -c util.c

main.o: main.c hello.h util.h

gcc -o main.o -c main.c





A Full Build

hello: hello.o util.o main.o
does hello.o exist? NO
is there a rule for hello.o? YES

hello.o: hello.c hello.h

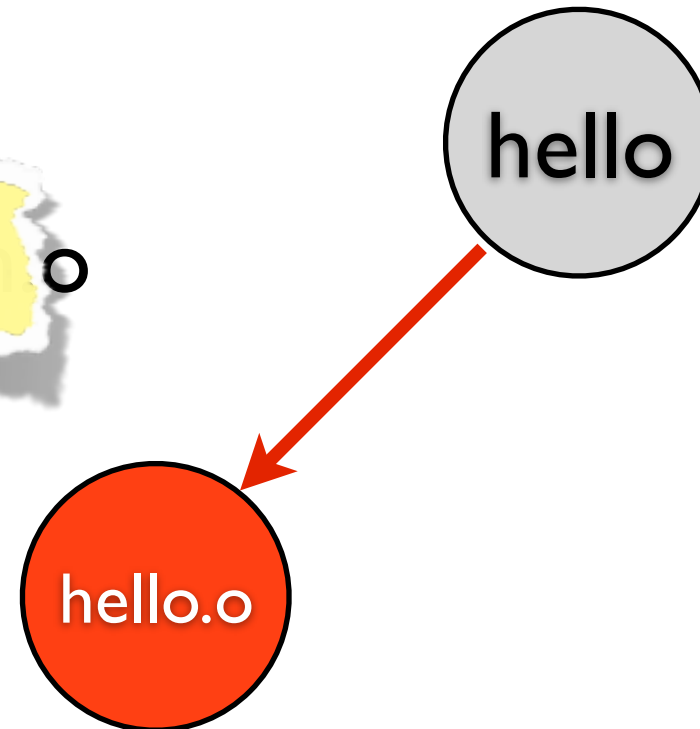
gcc -o hello.o -c hello.c

util.o: util.c util.h

gcc -o util.o -c util.c

main.o: main.c hello.h util.h

gcc -o main.o -c main.c





A Full Build

hello: hello.o util.o main.o

gcc -o hello hello.o util.o main.o

hello.o: **hello.c** hello.h

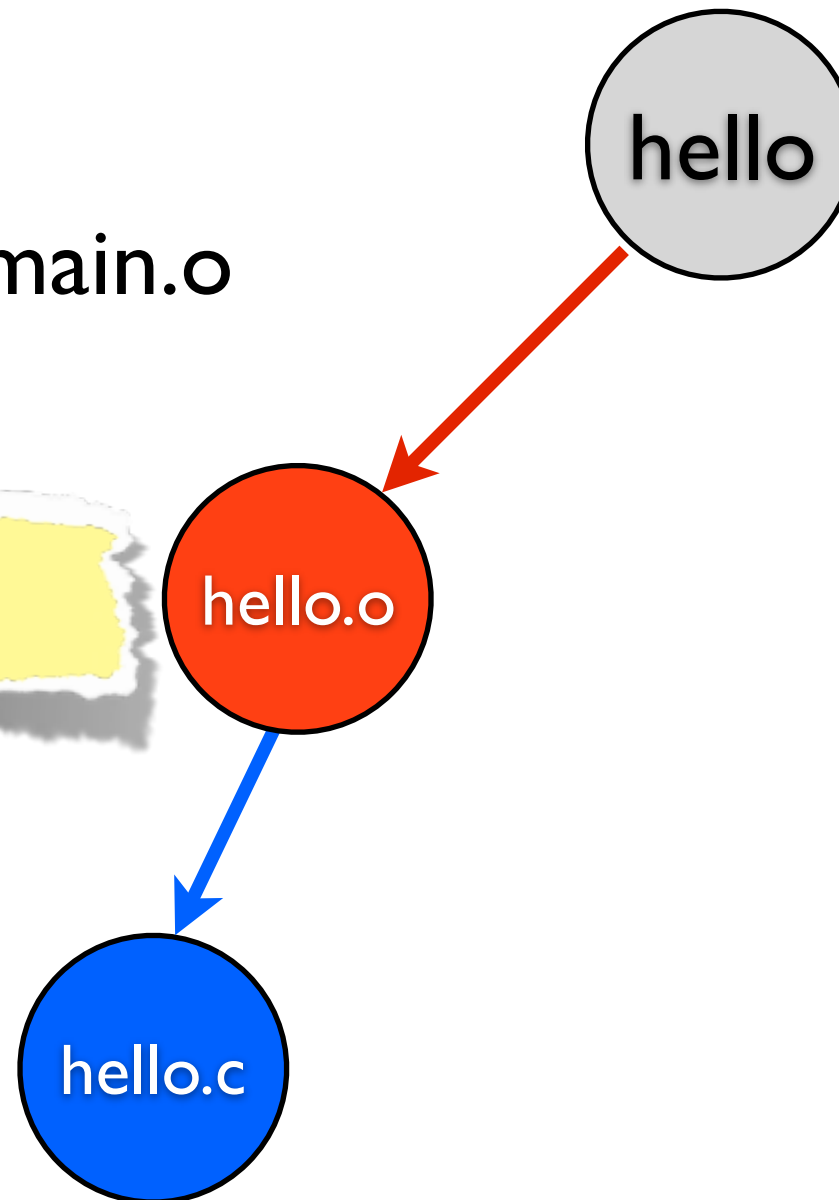
does hello.c exist? YES

util.o: util.c util.h

gcc -o util.o -c util.c

main.o: main.c hello.h util.h

gcc -o main.o -c main.c





A Full Build

hello: hello.o util.o main.o

gcc -o hello hello.o util.o main.o

hello.o: hello.c **hello.h**

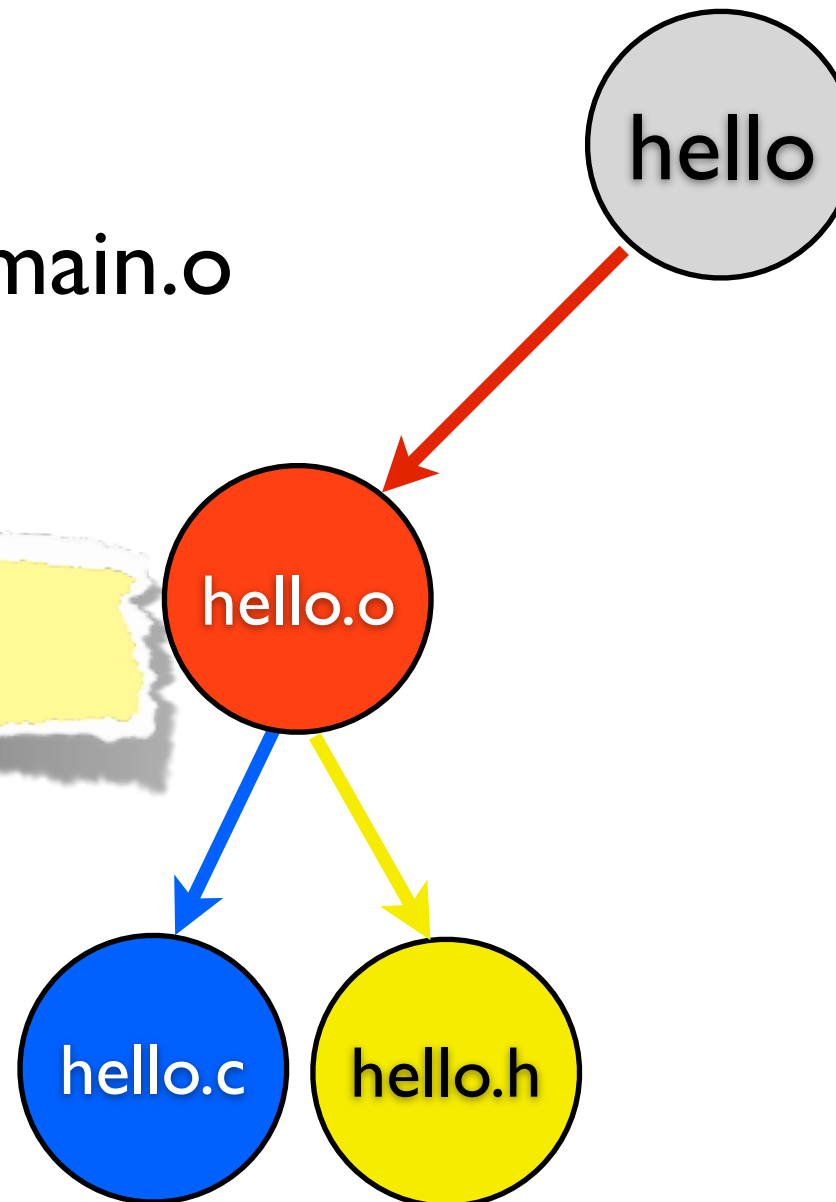
does hello.h exist? YES

util.o: util.c util.h

gcc -o util.o -c util.c

main.o: main.c hello.h util.h

gcc -o main.o -c main.c





A Full Build

hello: hello.o util.o main.o

gcc -o hello hello.o util.o main.o

hello.o: hello.c hello.h

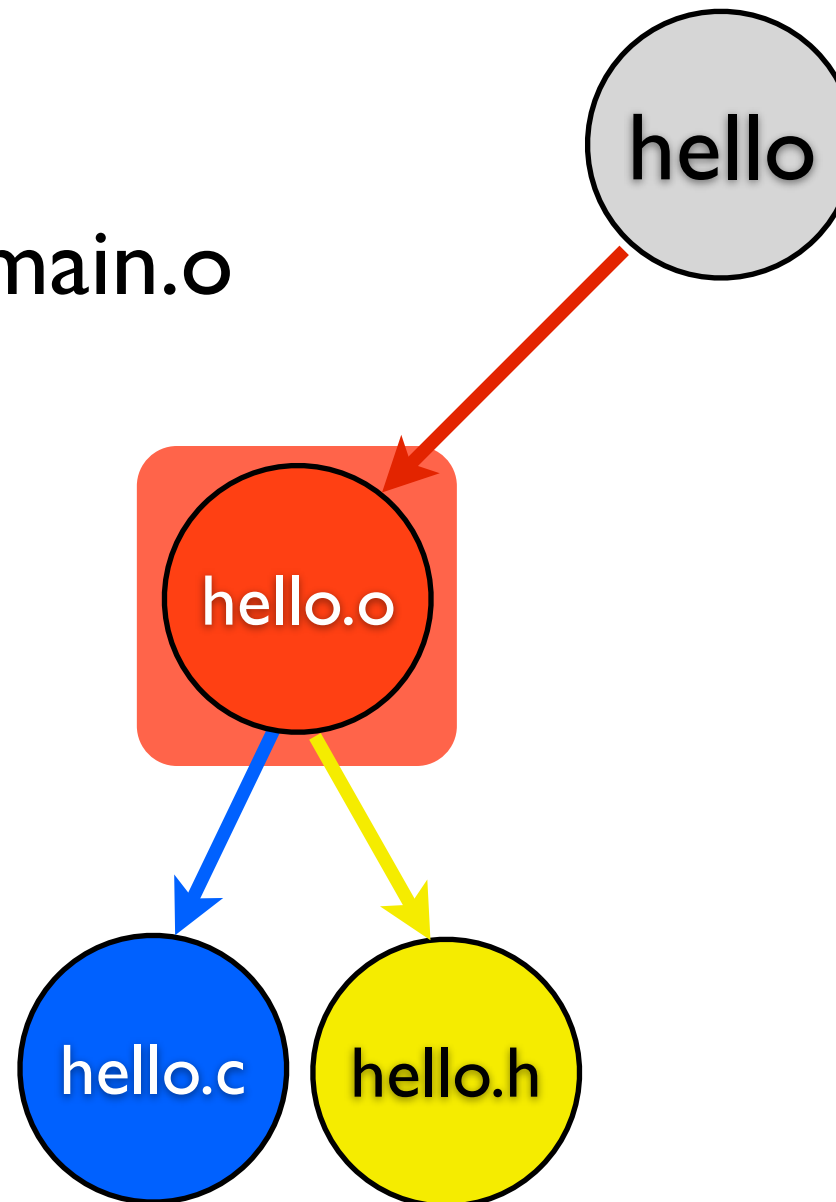
gcc -o hello.o -c hello.c

util.o: util.c util.h

gcc -o util.o -c util.c

main.o: main.c hello.h util.h

gcc -o main.o -c main.c





A Full Build

hello: hello.o **util.o** main.o

gcc -o hello hello.o util.o main.o

hello.o: hello.c hello.h

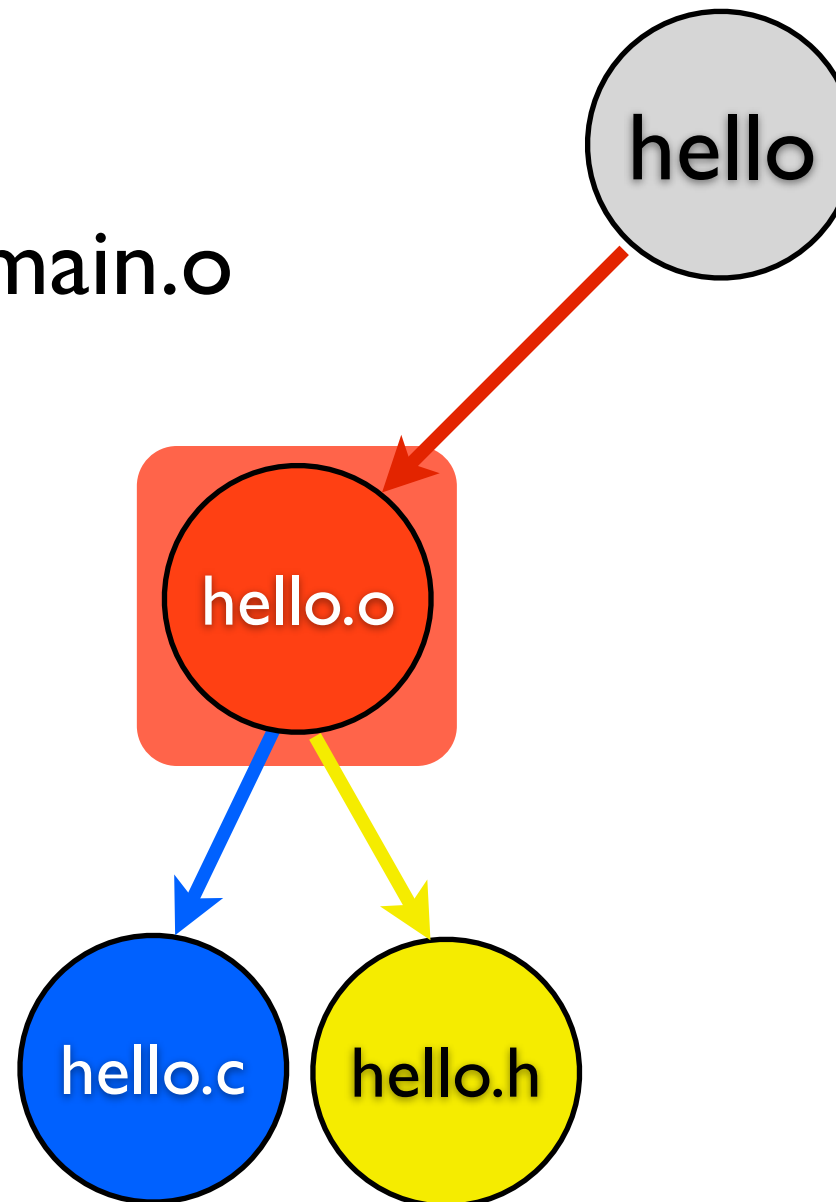
gcc -o hello.o -c hello.c

util.o: util.c util.h

gcc -o util.o -c util.c

main.o: main.c hello.h util.h

gcc -o main.o -c main.c





A Full Build

hello: hello.o **util.o** main.o

gcc -o hello hello.o util.o main.o

hello.o: hello.c hello.h

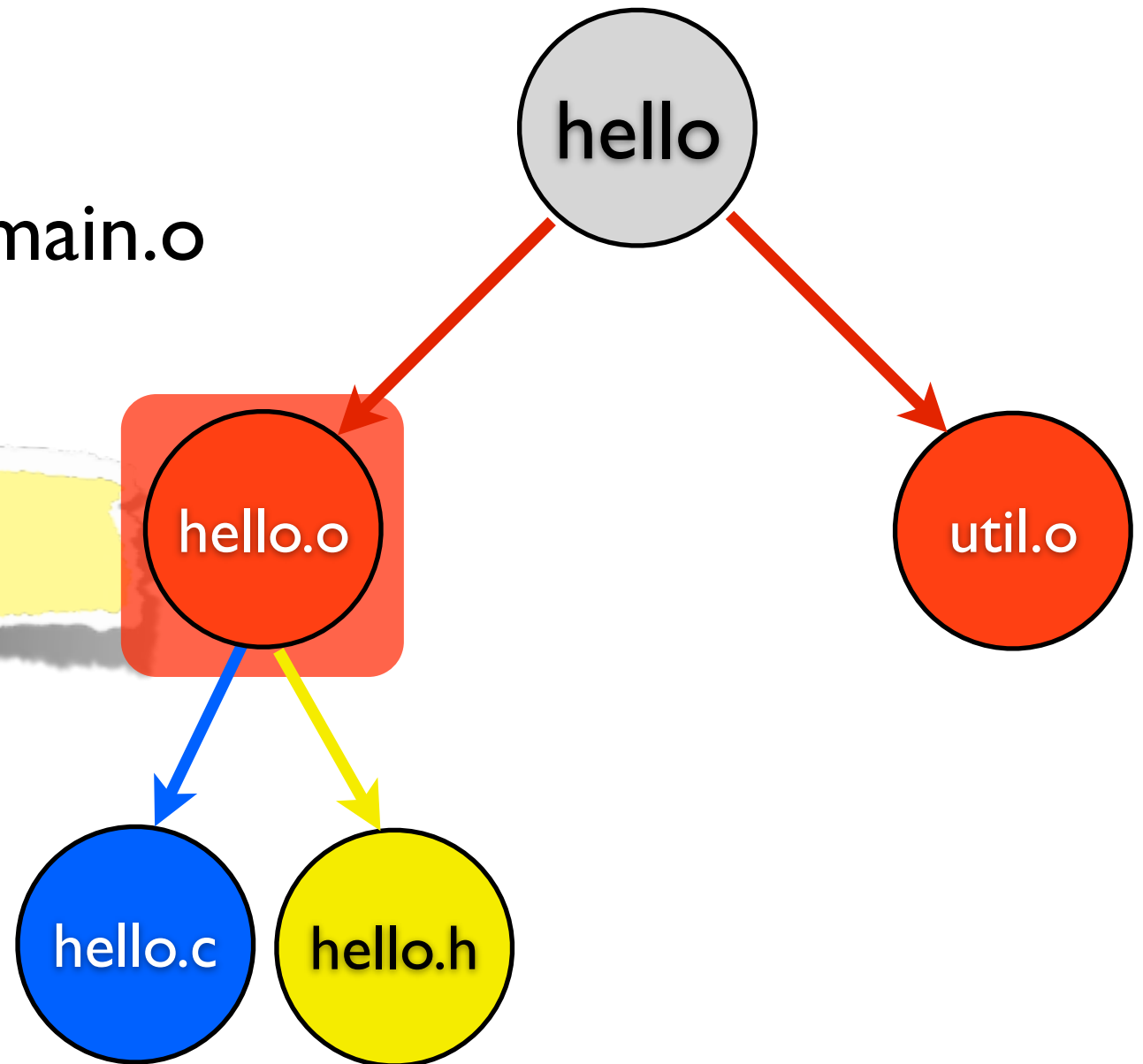
does util.o exist? NO
is there a rule for util.o? YES

util.o: util.c util.h

gcc -o util.o -c util.c

main.o: main.c hello.h util.h

gcc -o main.o -c main.c





A Full Build

hello: hello.o util.o main.o

gcc -o hello hello.o util.o main.o

hello.o: hello.c hello.h

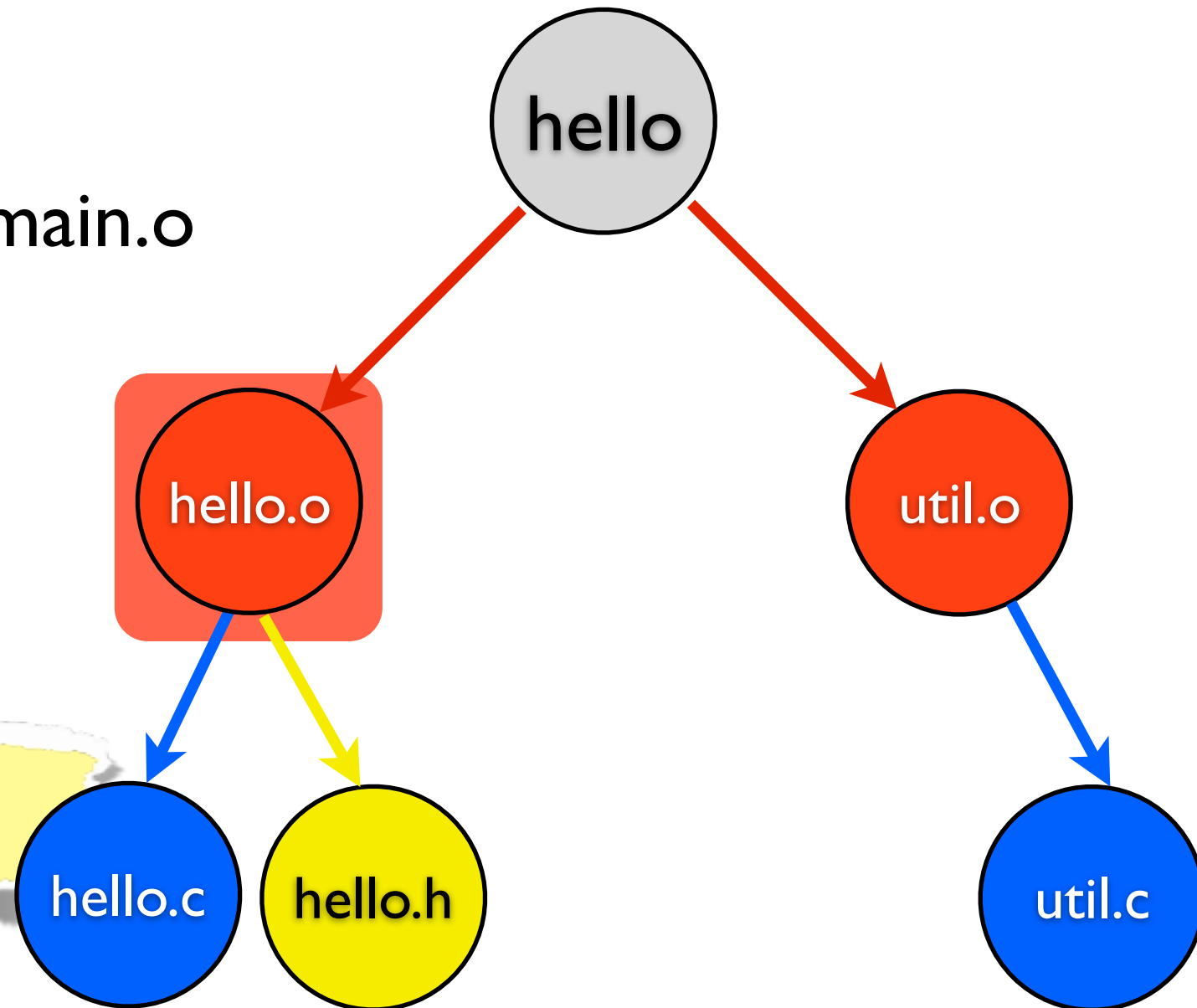
gcc -o hello.o -c hello.c

util.o: util.c util.h

does util.c exist? YES

main.o: main.c hello.h util.h

gcc -o main.o -c main.c





A Full Build

hello: hello.o util.o main.o

gcc -o hello hello.o util.o main.o

hello.o: hello.c hello.h

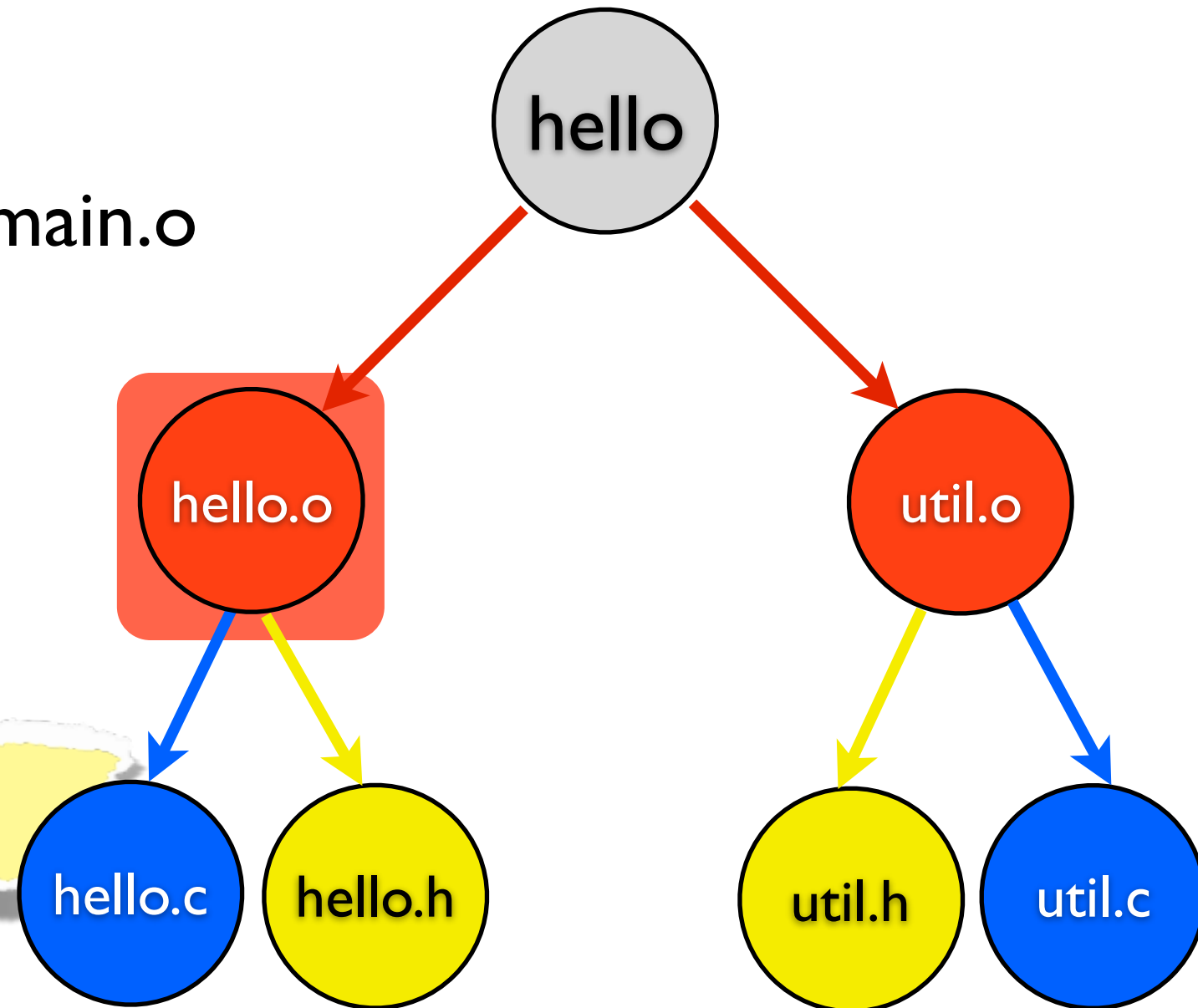
gcc -o hello.o -c hello.c

util.o: util.c **util.h**

does util.h exist? YES

main.o: main.c hello.h util.h

gcc -o main.o -c main.c





A Full Build

hello: hello.o util.o main.o

gcc -o hello hello.o util.o main.o

hello.o: hello.c hello.h

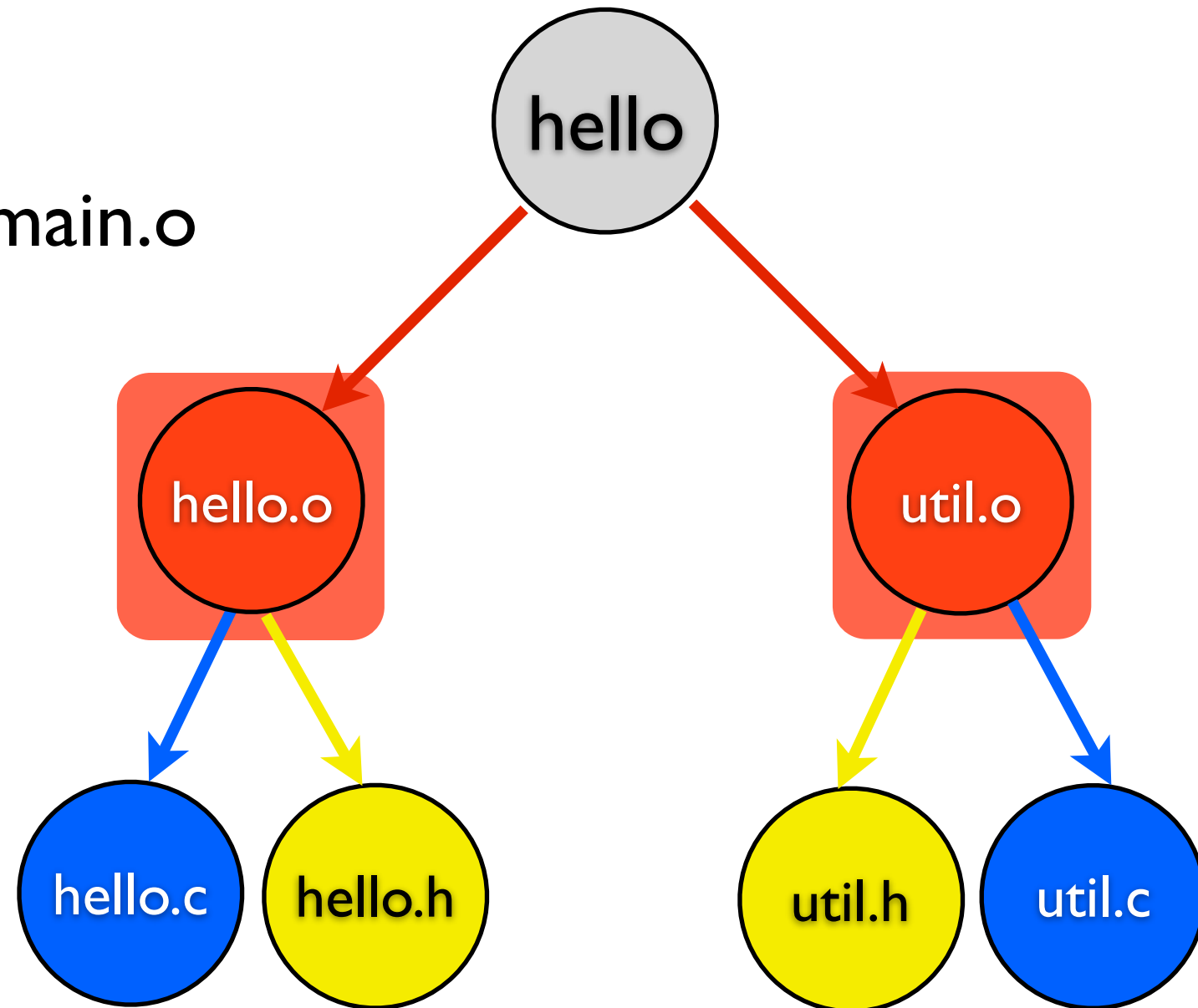
gcc -o hello.o -c hello.c

util.o: util.c util.h

gcc -o util.o -c util.c

main.o: main.c hello.h util.h

gcc -o main.o -c main.c





A Full Build

hello: hello.o util.o **main.o**

gcc -o hello hello.o util.o main.o

hello.o: hello.c hello.h

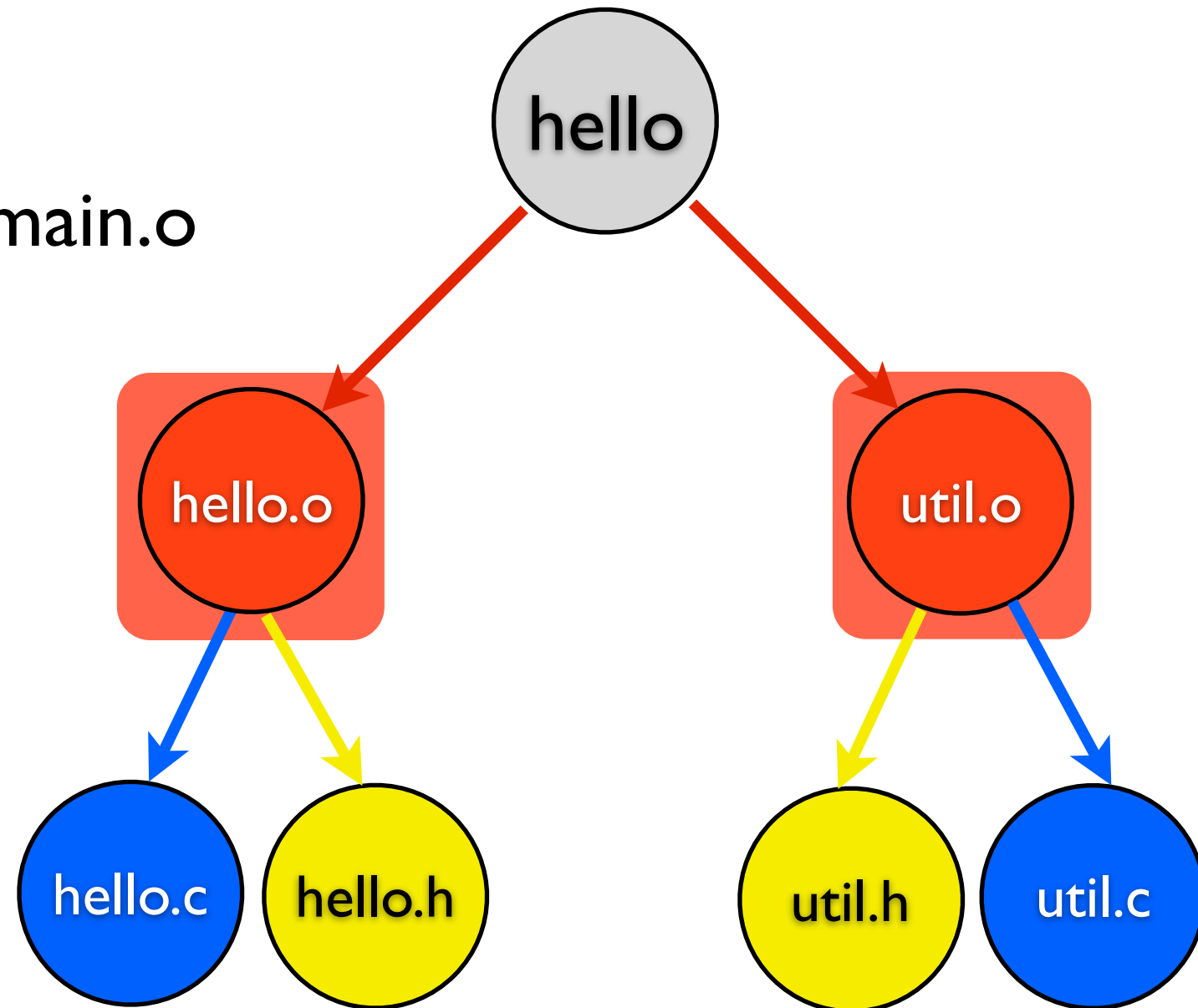
gcc -o hello.o -c hello.c

util.o: util.c util.h

gcc -o util.o -c util.c

main.o: main.c hello.h util.h

gcc -o main.o -c main.c





A Full Build

hello: hello.o util.o **main.o**

gcc -o hello hello.o util.o main.o

hello.o: hello.c hello.h

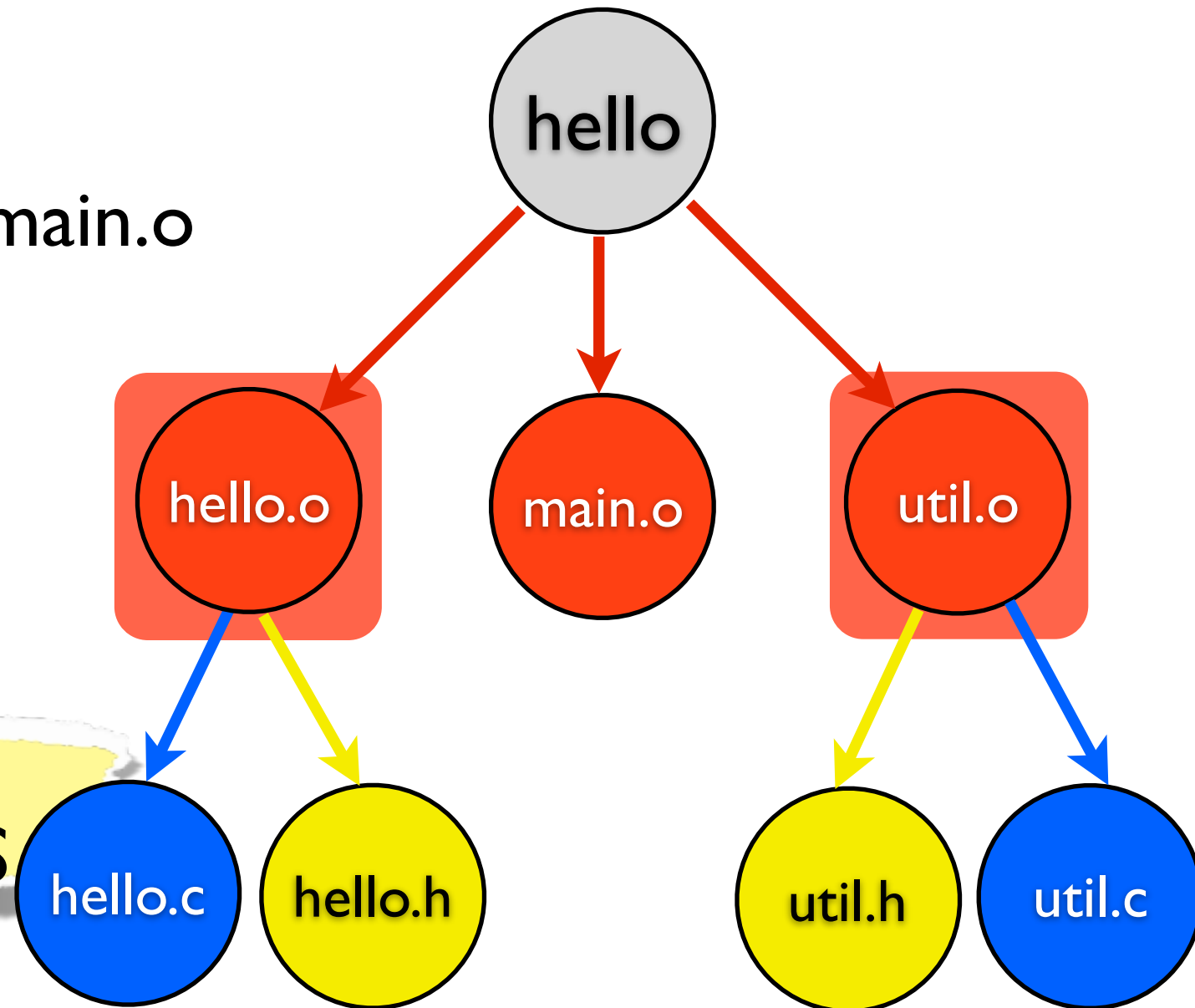
gcc -o hello.o -c hello.c

util.o: util.c util.h

does main.o exist? NO
is there a rule for main.o? YES

main.o main.c hello.h util.h

gcc -o main.o -c main.c





A Full Build

hello: hello.o util.o main.o

gcc -o hello hello.o util.o main.o

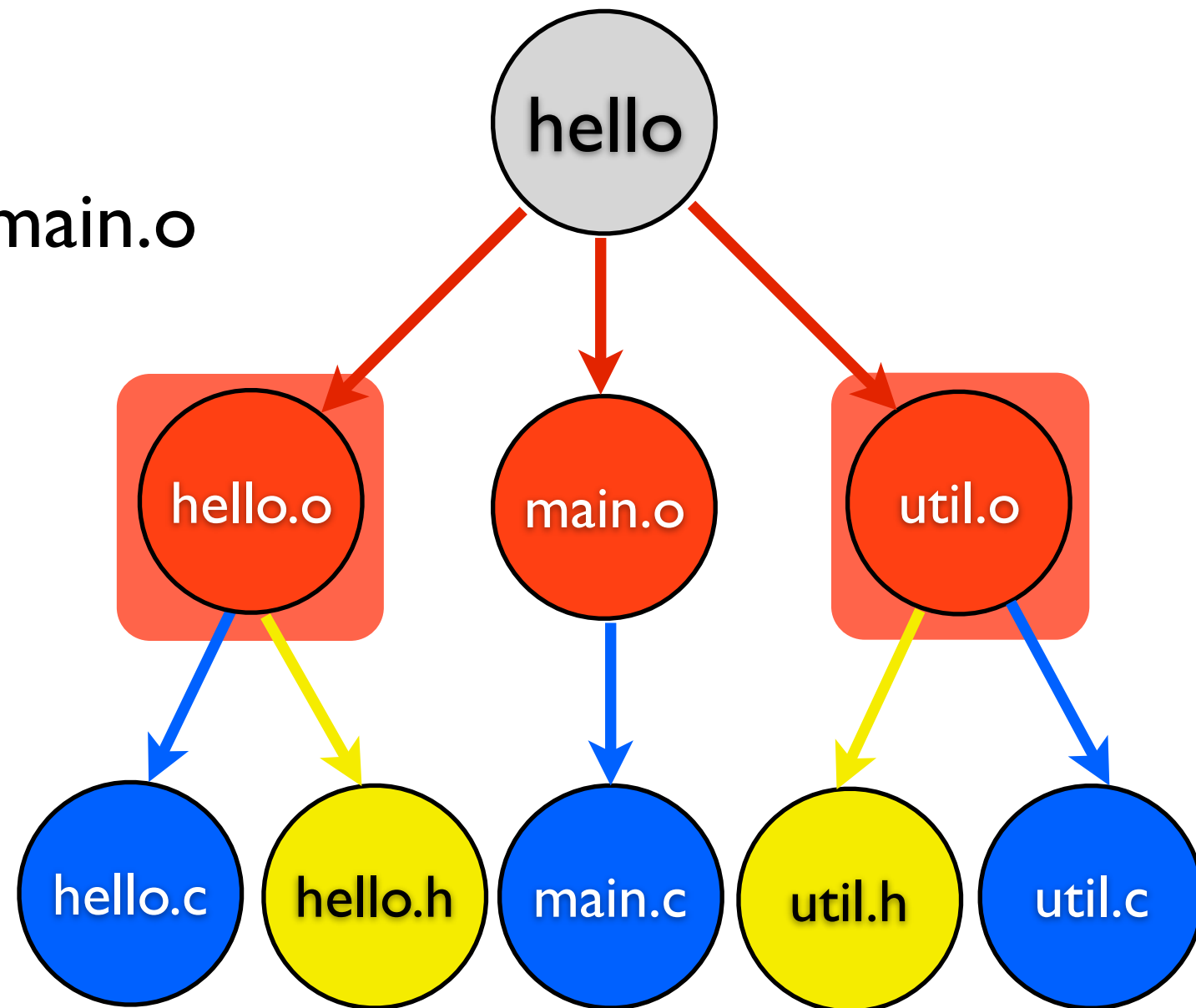
hello.o: hello.c hello.h

gcc -o hello.o -c hello.c

util.o: util.c util.h

gcc -o util.o -c util.c

main.o: **main.c** hello.h util.h



does main.c exist? YES



A Full Build

hello: hello.o util.o main.o

gcc -o hello hello.o util.o main.o

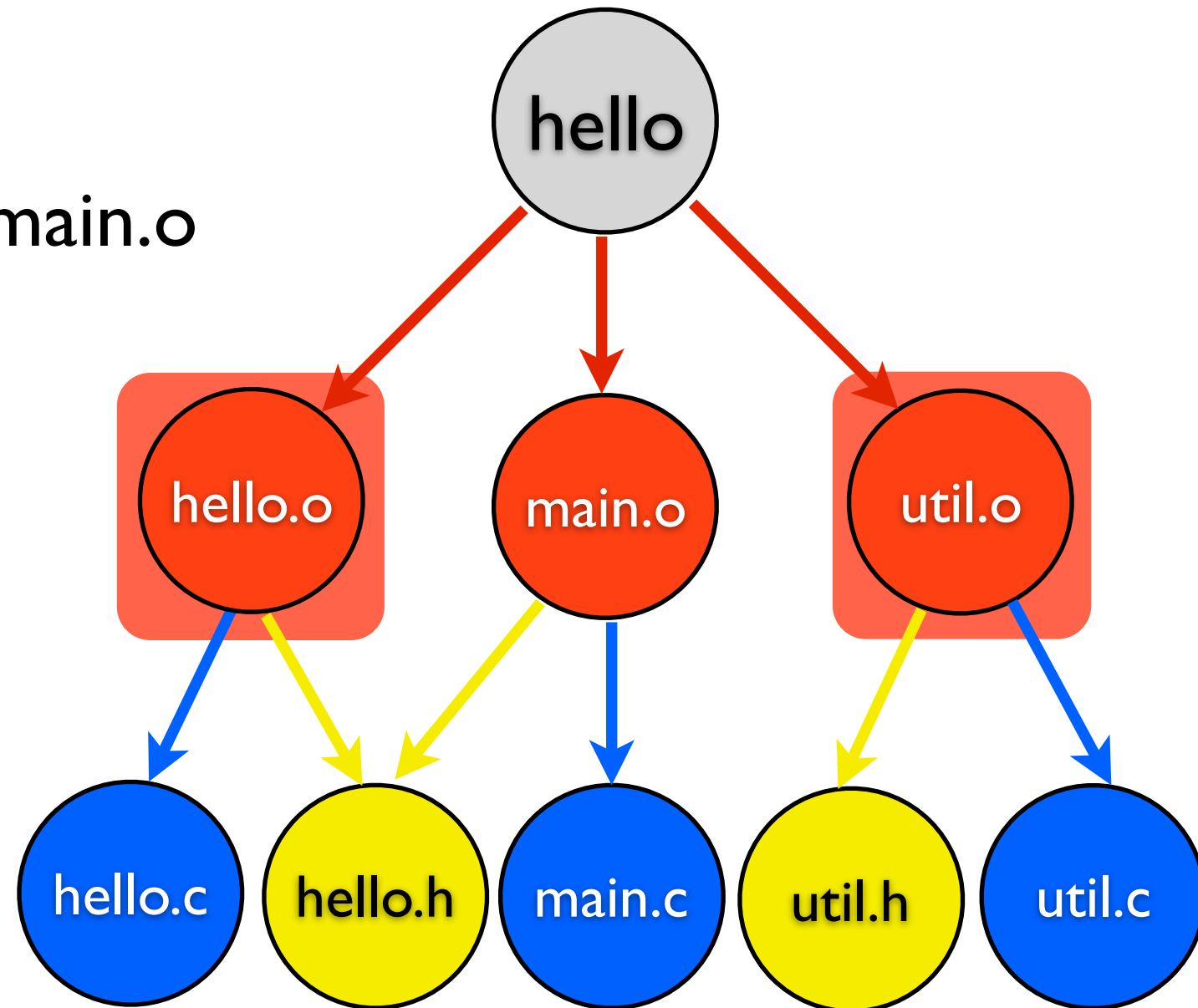
hello.o: hello.c hello.h

gcc -o hello.o -c hello.c

util.o: util.c util.h

gcc -o util.o -c util.c

main.o: main.c **hello.h** util.h



does hello.h exist? YES



A Full Build

hello: hello.o util.o main.o

gcc -o hello hello.o util.o main.o

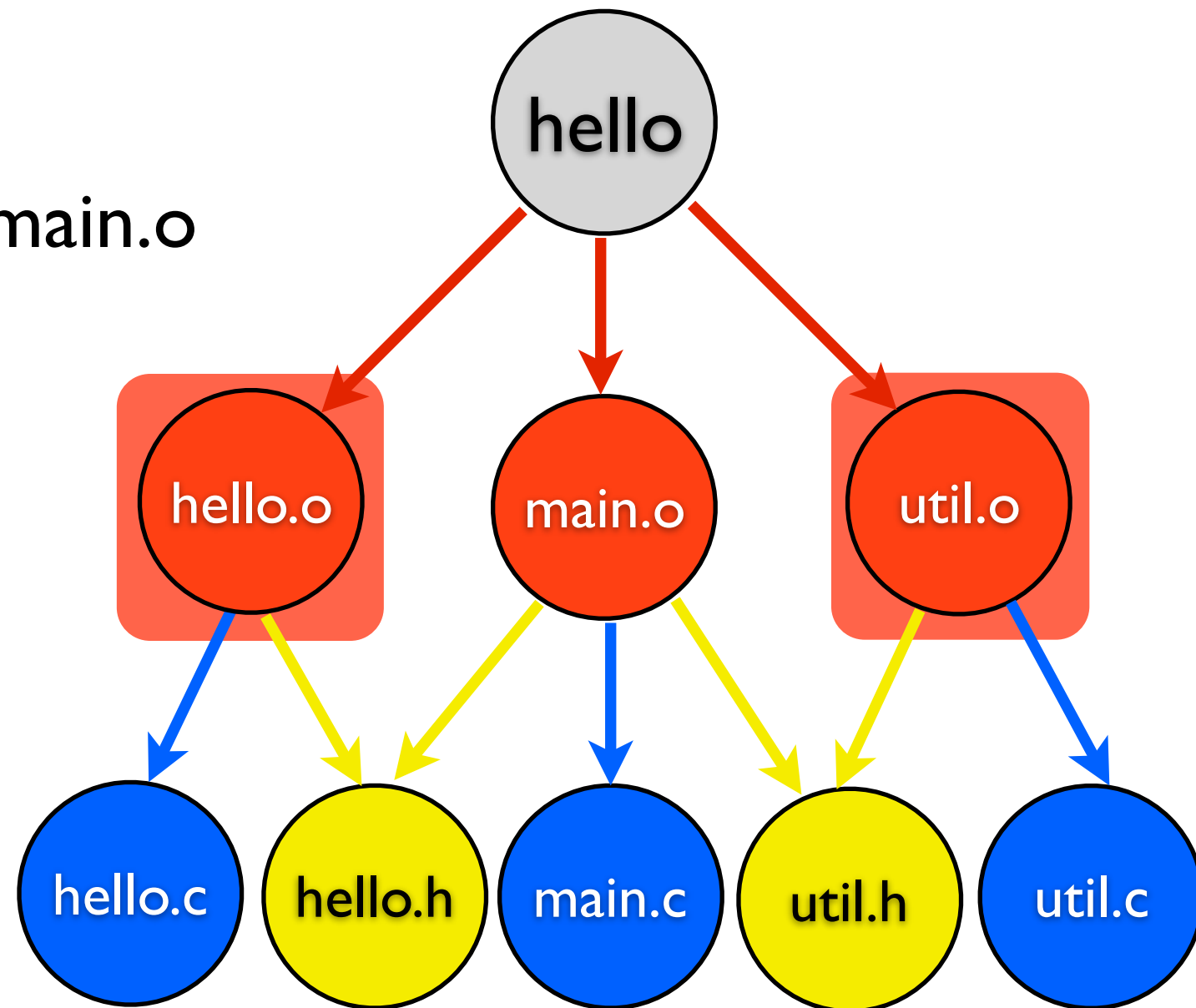
hello.o: hello.c hello.h

gcc -o hello.o -c hello.c

util.o: util.c util.h

gcc -o util.o -c util.c

main.o: main.c hello.h util.h



does util.h exist? YES



A Full Build

hello: hello.o util.o main.o

gcc -o hello hello.o util.o main.o

hello.o: hello.c hello.h

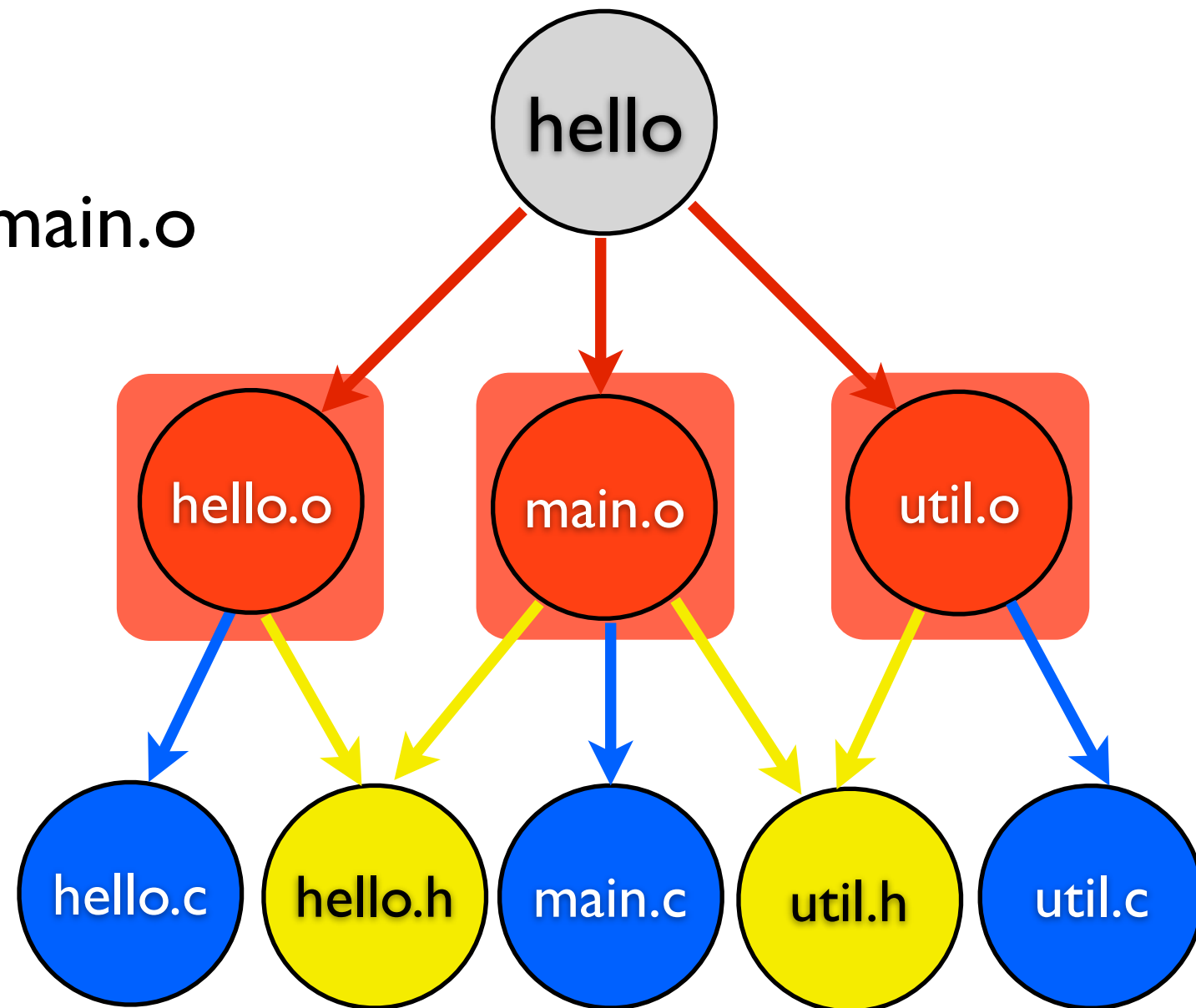
gcc -o hello.o -c hello.c

util.o: util.c util.h

gcc -o util.o -c util.c

main.o: main.c hello.h util.h

gcc -o main.o -c main.c





A Full Build

hello: hello.o util.o main.o

`gcc -o hello hello.o util.o main.o`

hello.o: hello.c hello.h

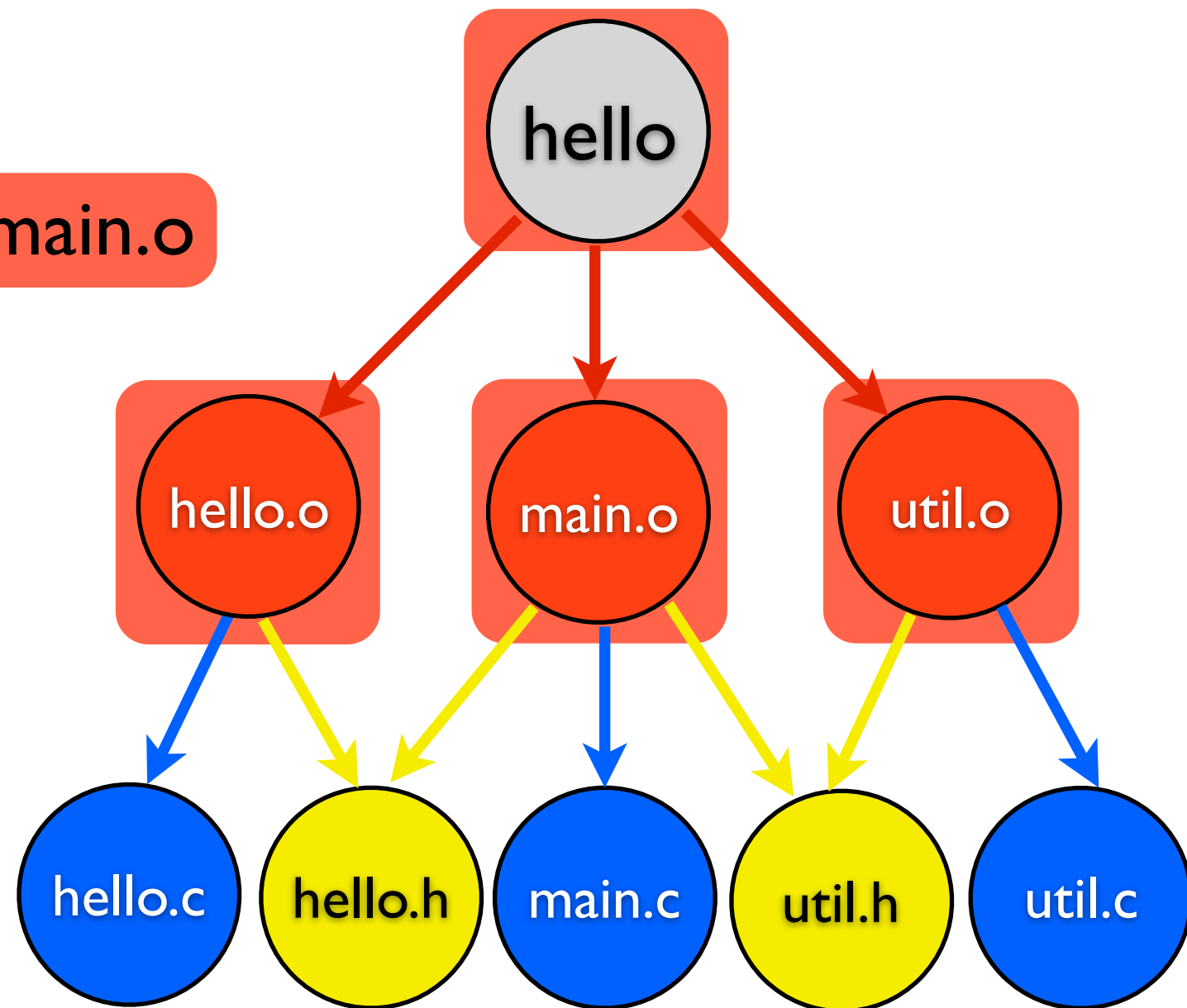
`gcc -o hello.o -c hello.c`

util.o: util.c util.h

`gcc -o util.o -c util.c`

main.o: main.c hello.h util.h

`gcc -o main.o -c main.c`





A Full Build

hello: hello.o util.o main.o

gcc -o hello hello.o util.o main.o

hello.o: hello.c hello.h

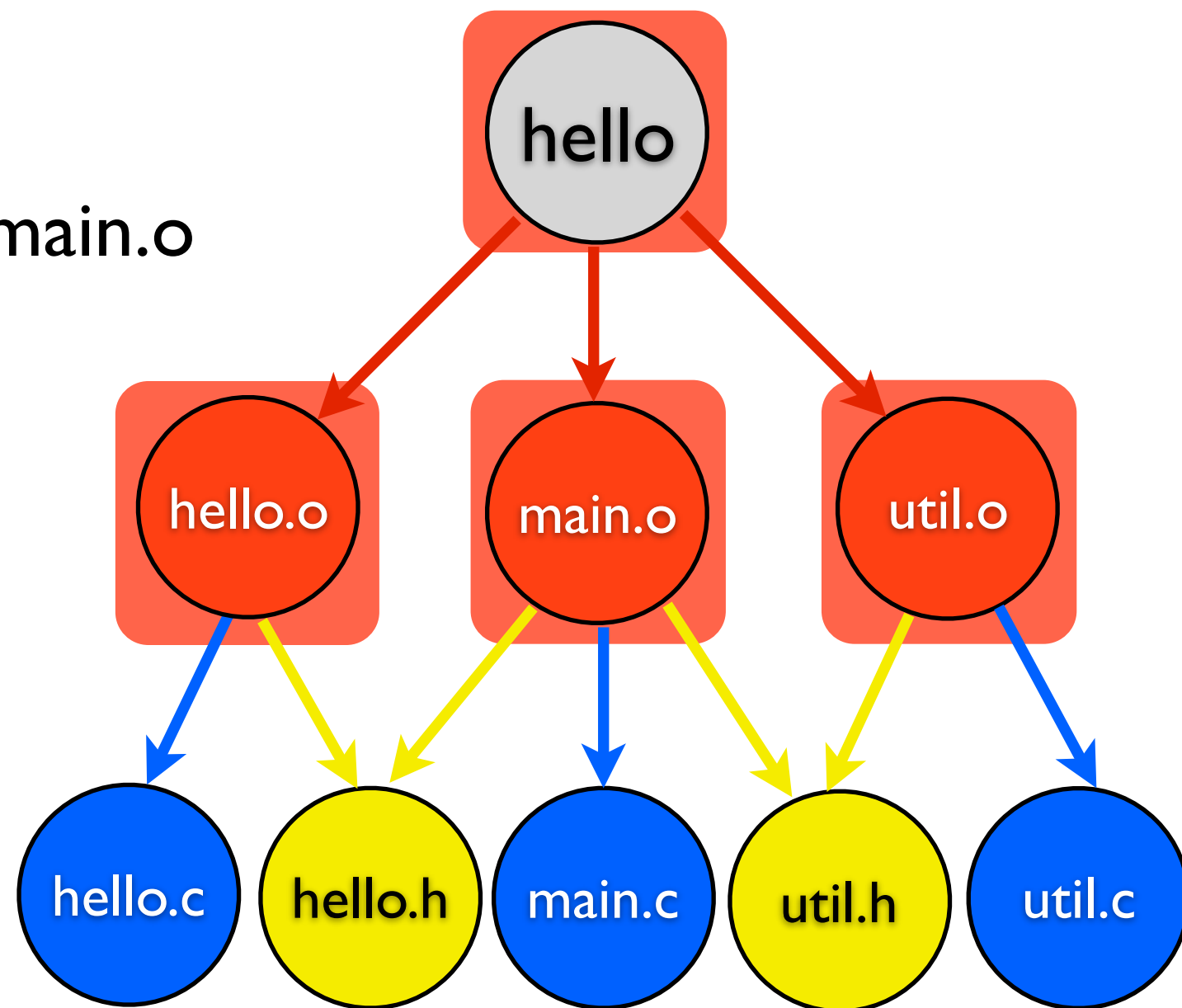
gcc -o hello.o -c hello.c

util.o: util.c util.h

gcc -o util.o -c util.c

main.o: main.c hello.h util.h

gcc -o main.o -c main.c



4 build commands
executed!

Yeah right, I
could do
that with
Bash as
well ...





An incremental build after changing hello.h

hello: hello.o util.o main.o

gcc -o hello hello.o util.o main.o

hello.o: hello.c hello.h

gcc -o hello.o -c hello.c

util.o: util.c util.h

gcc -o util.o -c util.c

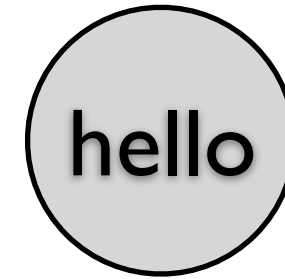
main.o: main.c hello.h util.h

gcc -o main.o -c main.c

An incremental build after changing hello.h



hello:hello.o util.o main.o



gcc -o hello hello.o util.o main.o

hello.o: hello.c hello.h

gcc -o hello.o -c hello.c

util.o: util.c util.h

gcc -o util.o -c util.c

main.o: main.c hello.h util.h

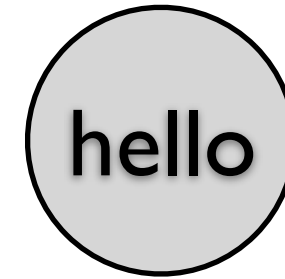
gcc -o main.o -c main.c

An incremental build after changing hello.h



hello: **hello.o** util.o main.o

gcc -o hello hello.o util.o main.o



hello.o: hello.c hello.h

gcc -o hello.o -c hello.c

util.o: util.c util.h

gcc -o util.o -c util.c

main.o: main.c hello.h util.h

gcc -o main.o -c main.c

An incremental build after changing hello.h



does hello.o exist? YES

is hello.o newer than all its dependencies?

hello.o: hello.c hello.h

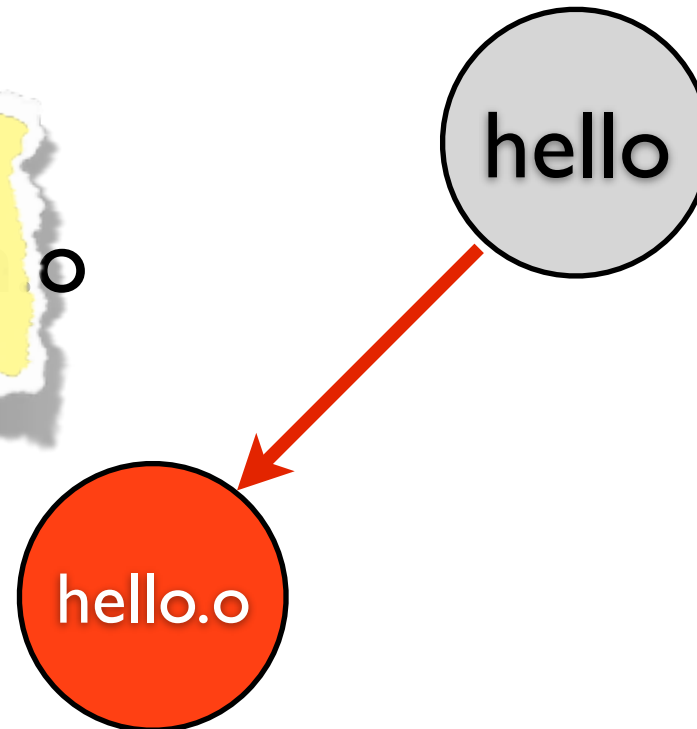
gcc -o hello.o -c hello.c

util.o: util.c util.h

gcc -o util.o -c util.c

main.o: main.c hello.h util.h

gcc -o main.o -c main.c



An incremental build after changing hello.h



does hello.o exist? YES
is hello.o newer than all its dependencies?

hello.o: hello.c hello.h

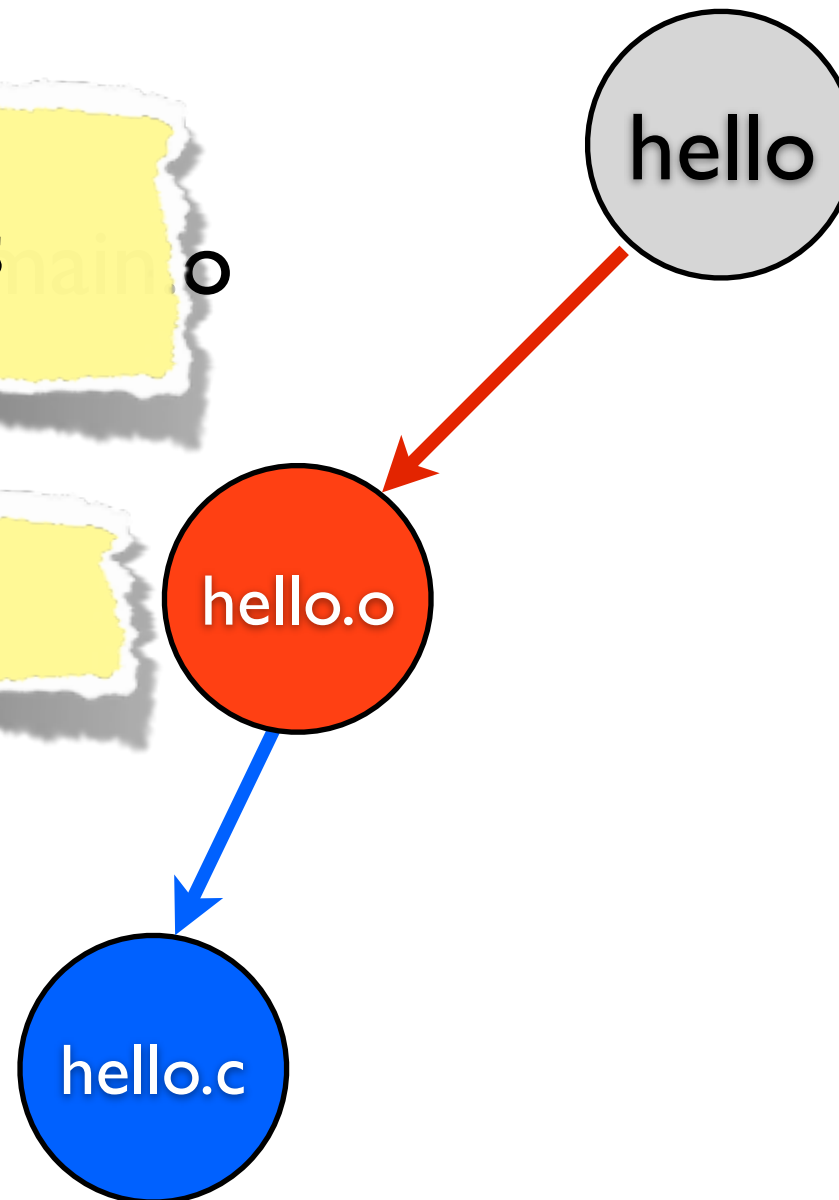
does hello.c exist? YES

util.o: util.c util.h

gcc -o util.o -c util.c

main.o: main.c hello.h util.h

gcc -o main.o -c main.c



An incremental build after changing hello.h



does hello.o exist? YES
is hello.o newer than all its dependencies?

hello.o: hello.c hello.h

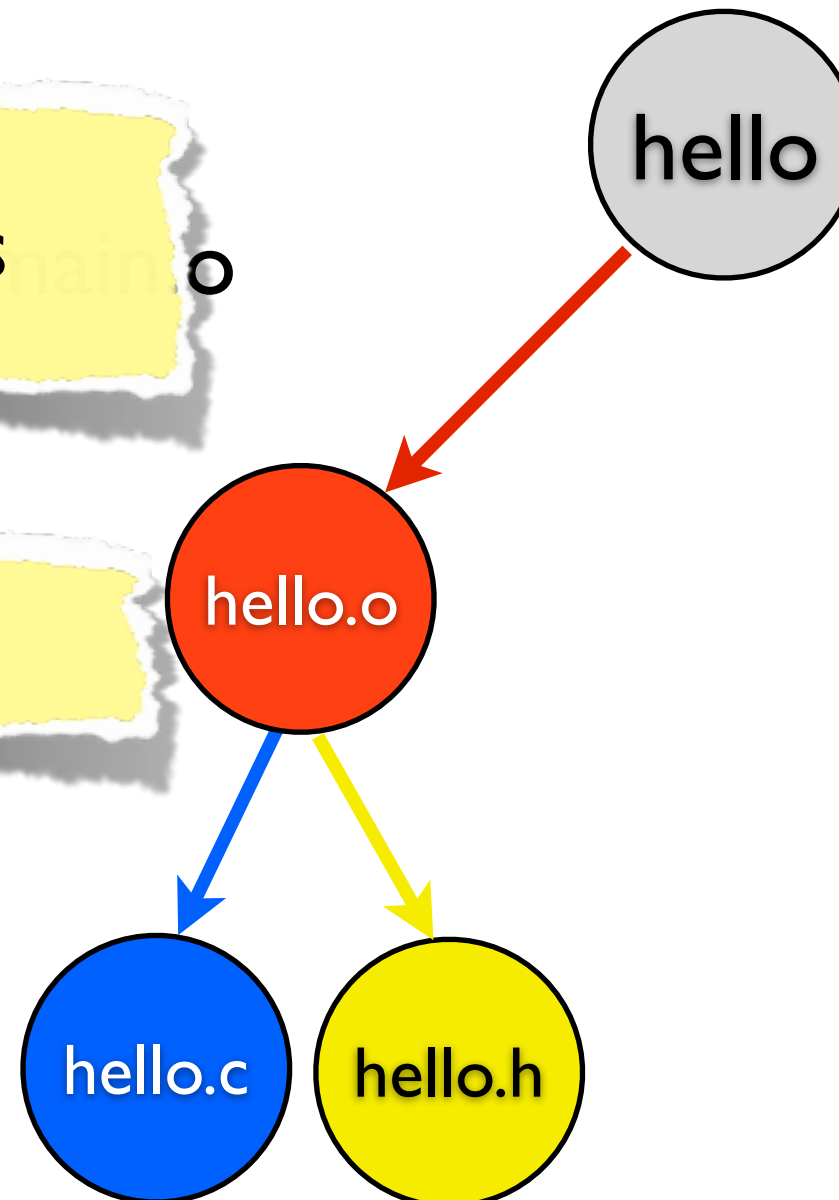
does hello.h exist? YES

util.o: util.c util.h

gcc -o util.o -c util.c

main.o: main.c hello.h util.h

gcc -o main.o -c main.c



An incremental build after changing hello.h



does hello.o exist? YES
is hello.o newer than all its dependencies? **NO**
hello.o: hello.c hello.h

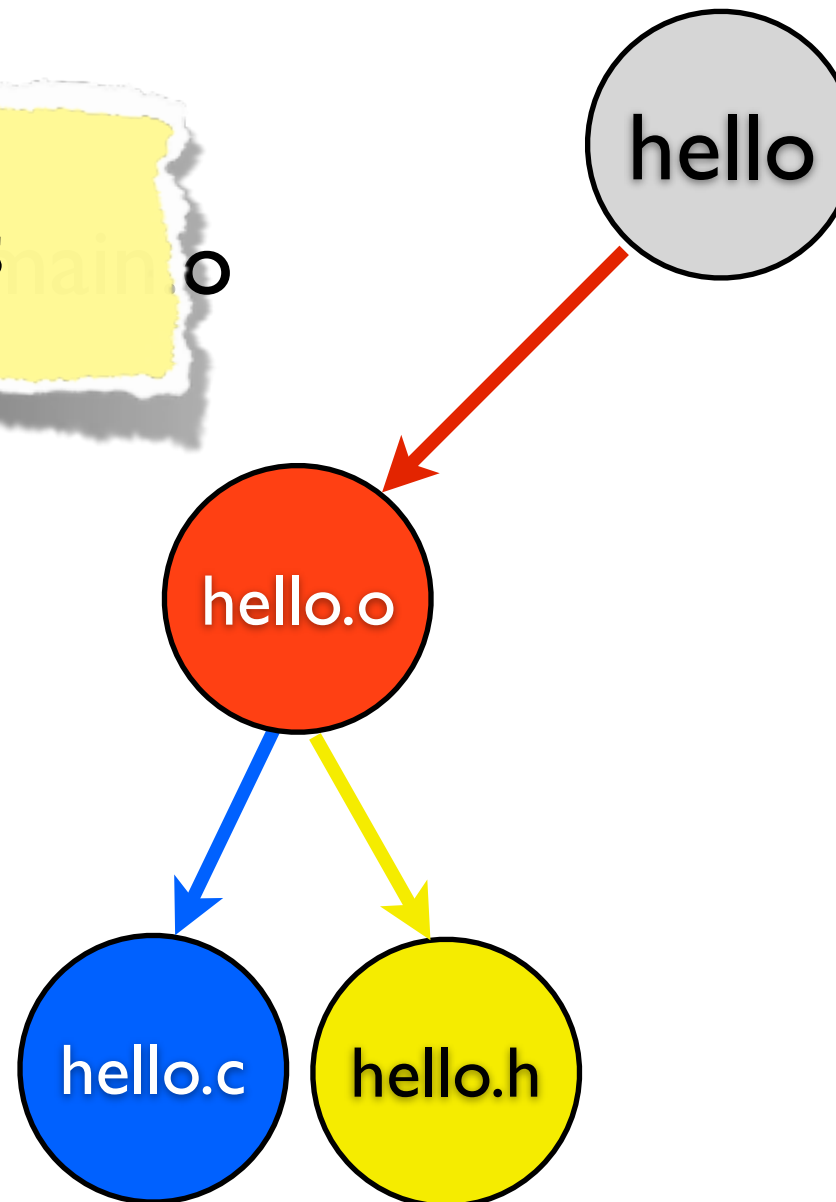
gcc -o hello.o -c hello.c

util.o: util.c util.h

gcc -o util.o -c util.c

main.o: main.c hello.h util.h

gcc -o main.o -c main.c





An incremental build after changing hello.h

hello: hello.o util.o main.o

gcc -o hello hello.o util.o main.o

hello.o: hello.c hello.h

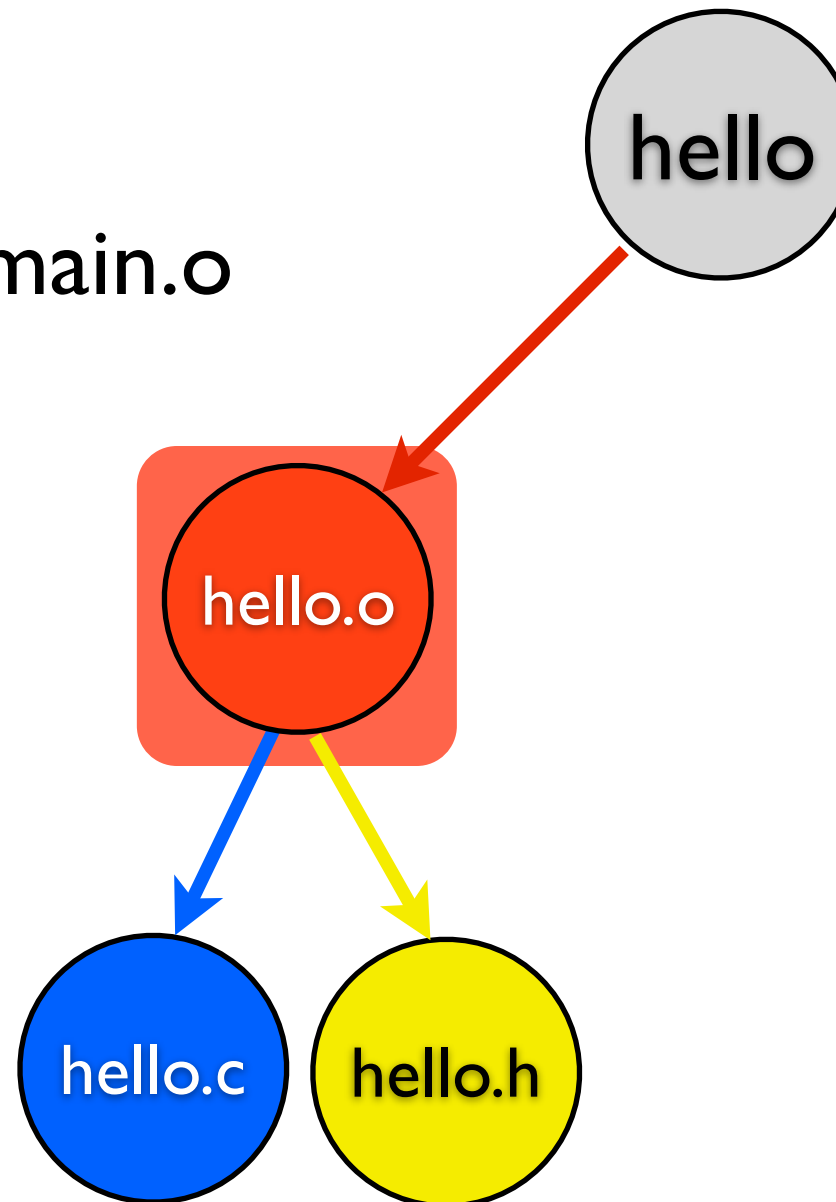
gcc -o hello.o -c hello.c

util.o: util.c util.h

gcc -o util.o -c util.c

main.o: main.c hello.h util.h

gcc -o main.o -c main.c





An incremental build after changing hello.h

hello: hello.o **util.o** main.o

gcc -o hello hello.o util.o main.o

hello.o: hello.c hello.h

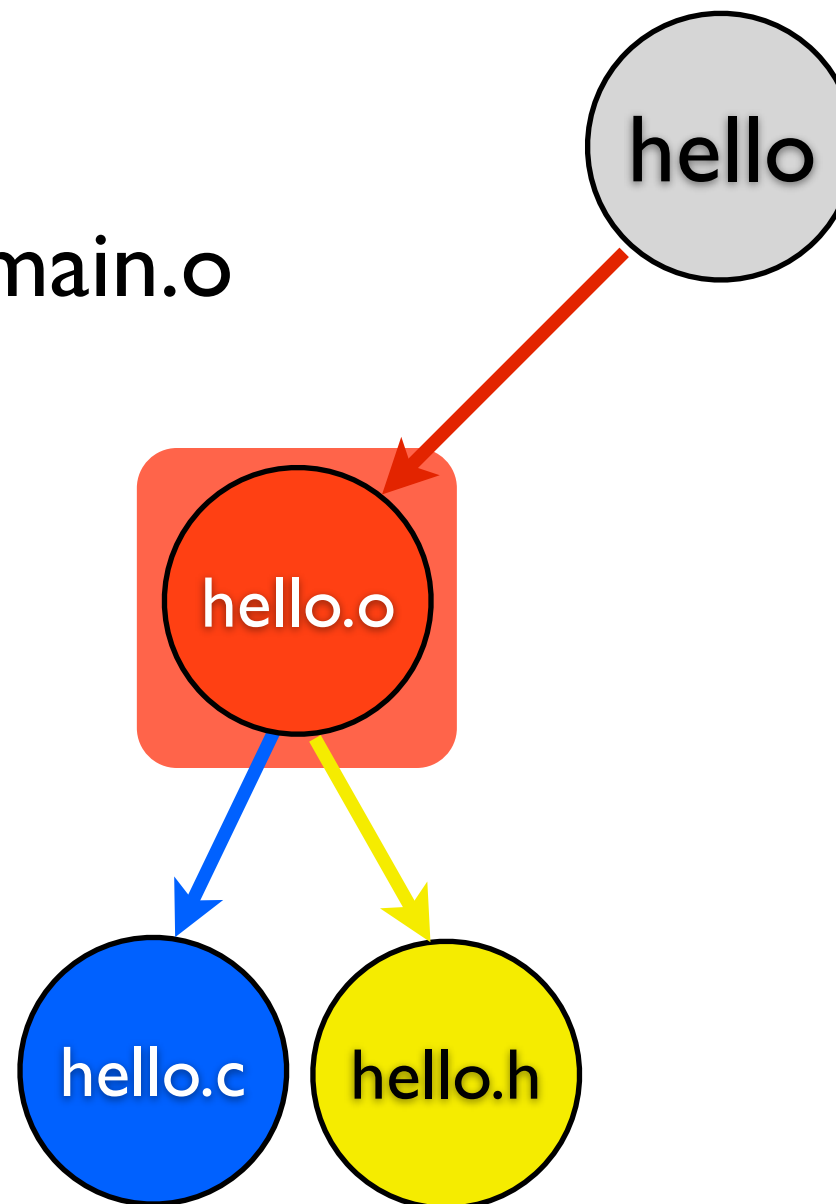
gcc -o hello.o -c hello.c

util.o: util.c util.h

gcc -o util.o -c util.c

main.o: main.c hello.h util.h

gcc -o main.o -c main.c





An incremental build after changing hello.h

hello: hello.o **util.o** main.o

gcc -o hello hello.o util.o main.o

hello.c: hello.c hello.h

does util.o exist? YES

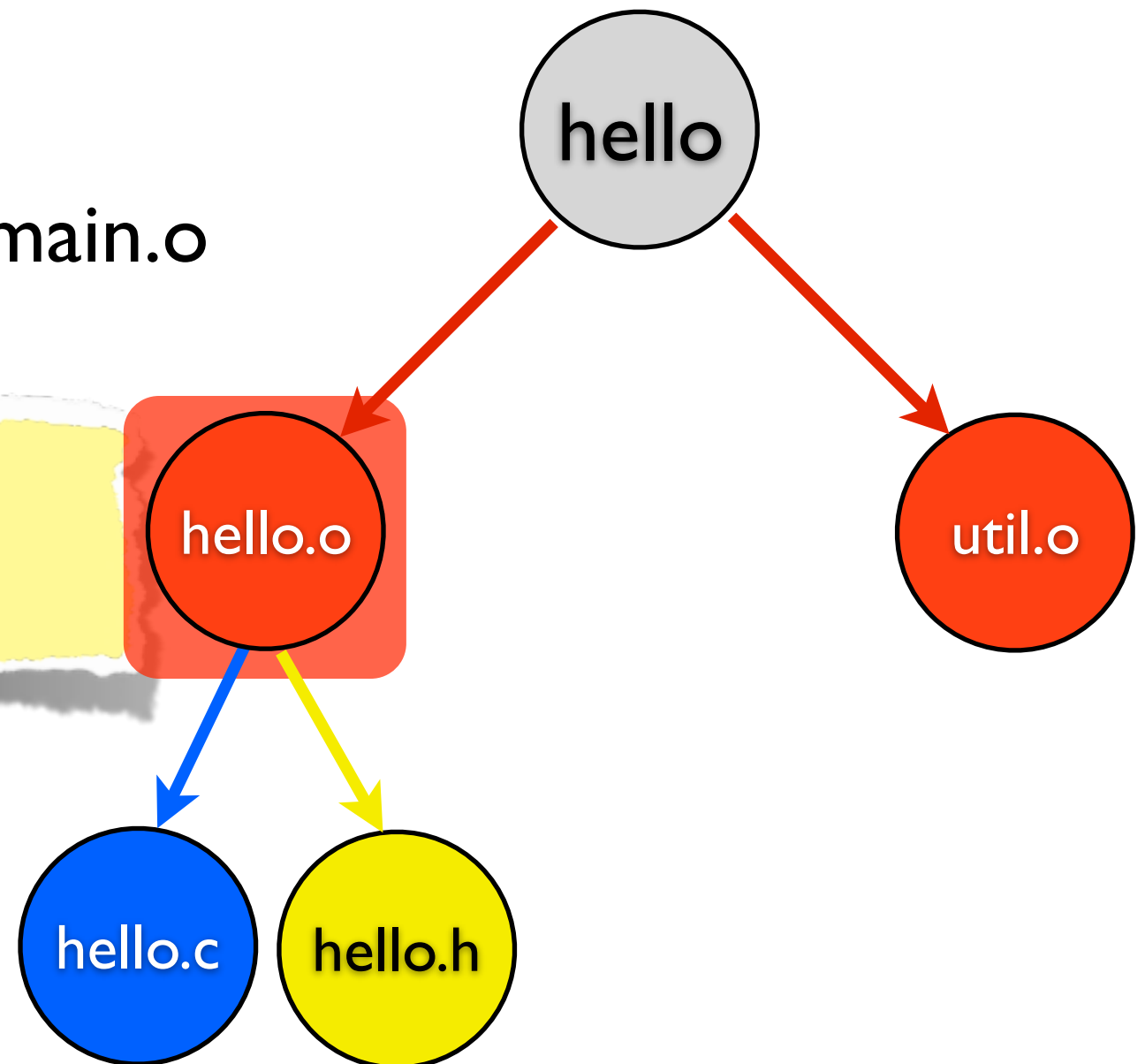
is util.o newer than all its dependencies?

util.o: util.c util.h

gcc -o util.o -c util.c

main.o: main.c hello.h util.h

gcc -o main.o -c main.c





An incremental build after changing hello.h

hello: hello.o util.o main.o

gcc -o hello hello.o util.o main.o

hello.c: hello.c hello.h

does util.o exist? YES

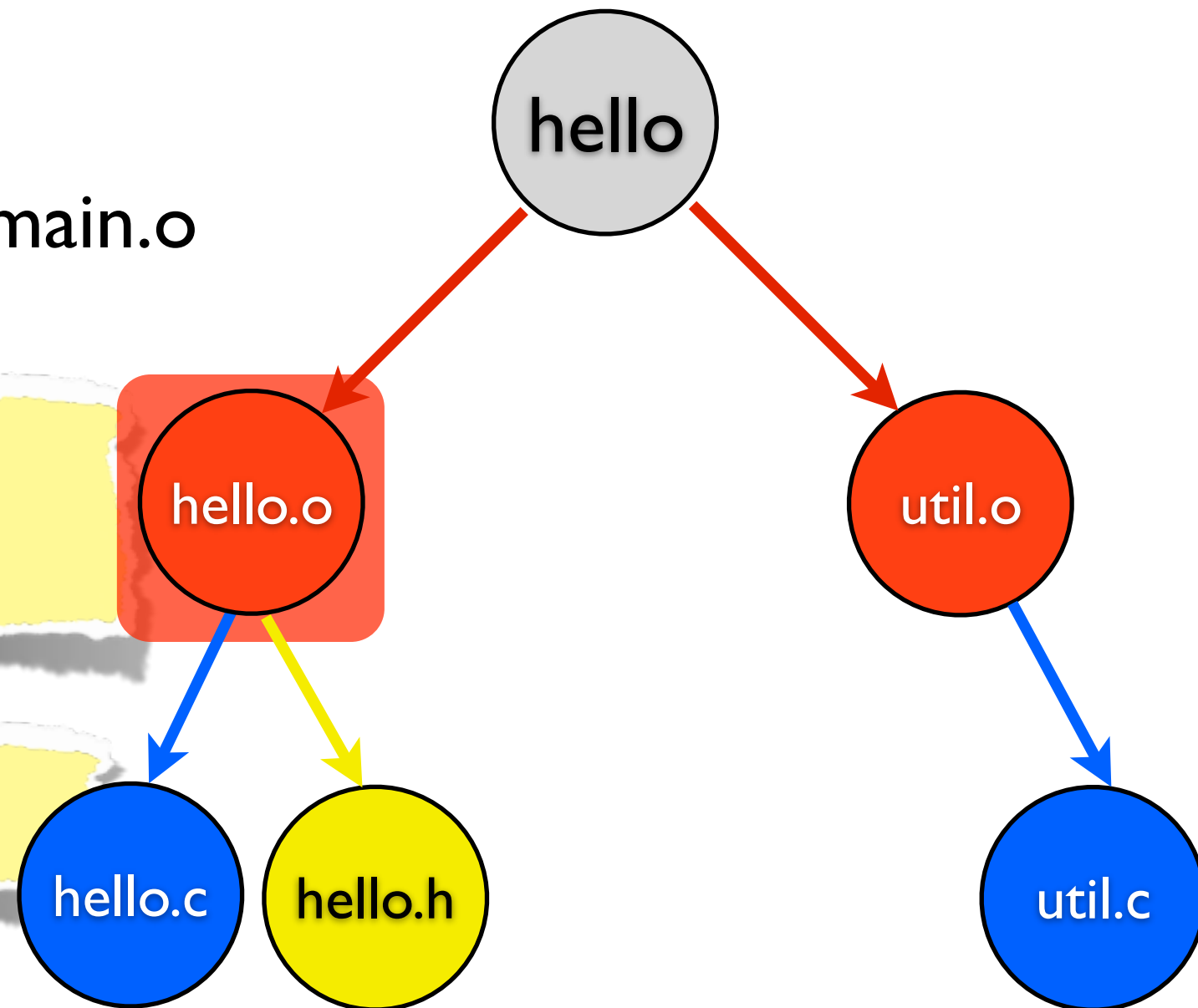
is util.o newer than all its dependencies?

util.o: util.c util.h

does util.c exist? YES

main.o: main.c hello.h util.h

gcc -o main.o -c main.c



An incremental build after changing hello.h

hello: hello.o util.o main.o

gcc -o hello hello.o util.o main.o

hello.c: hello.c hello.h

does util.o exist? YES

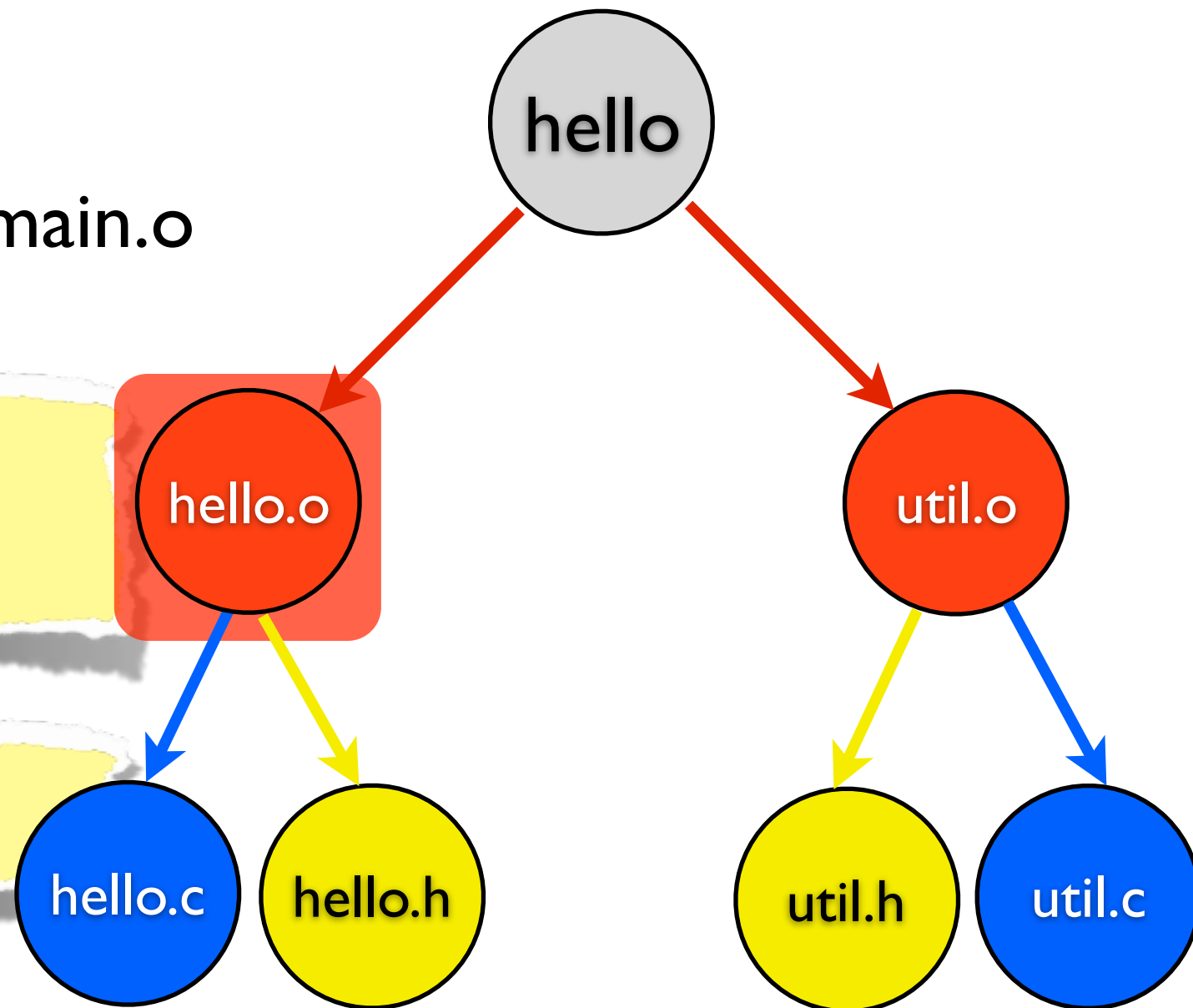
is util.o newer than all its dependencies?

util.o: util.c util.h

does util.h exist? YES

main.o: main.c hello.h util.h

gcc -o main.o -c main.c



An incremental build after changing hello.h

hello: hello.o util.o main.o

gcc -o hello hello.o util.o main.o

util.o: util.c util.h

does util.o exist? YES

is util.o newer than all its

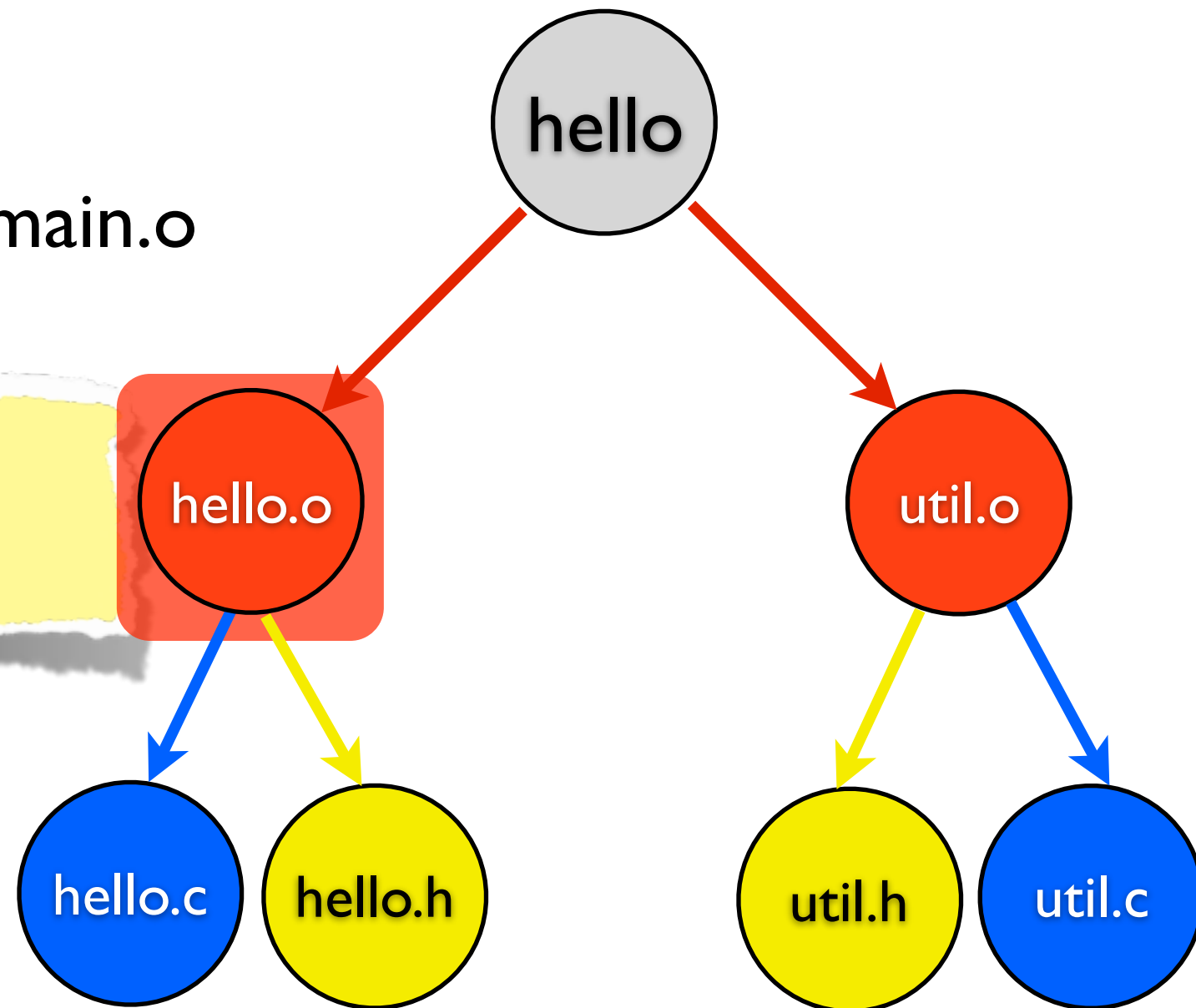
dependencies? **YES!**

util.o: util.c util.h

gcc -o util.o -c util.c

main.o: main.c hello.h util.h

gcc -o main.o -c main.c





An incremental build after changing hello.h

hello: hello.o util.o main.o

gcc -o hello hello.o util.o main.o

hello.o: hello.c hello.h

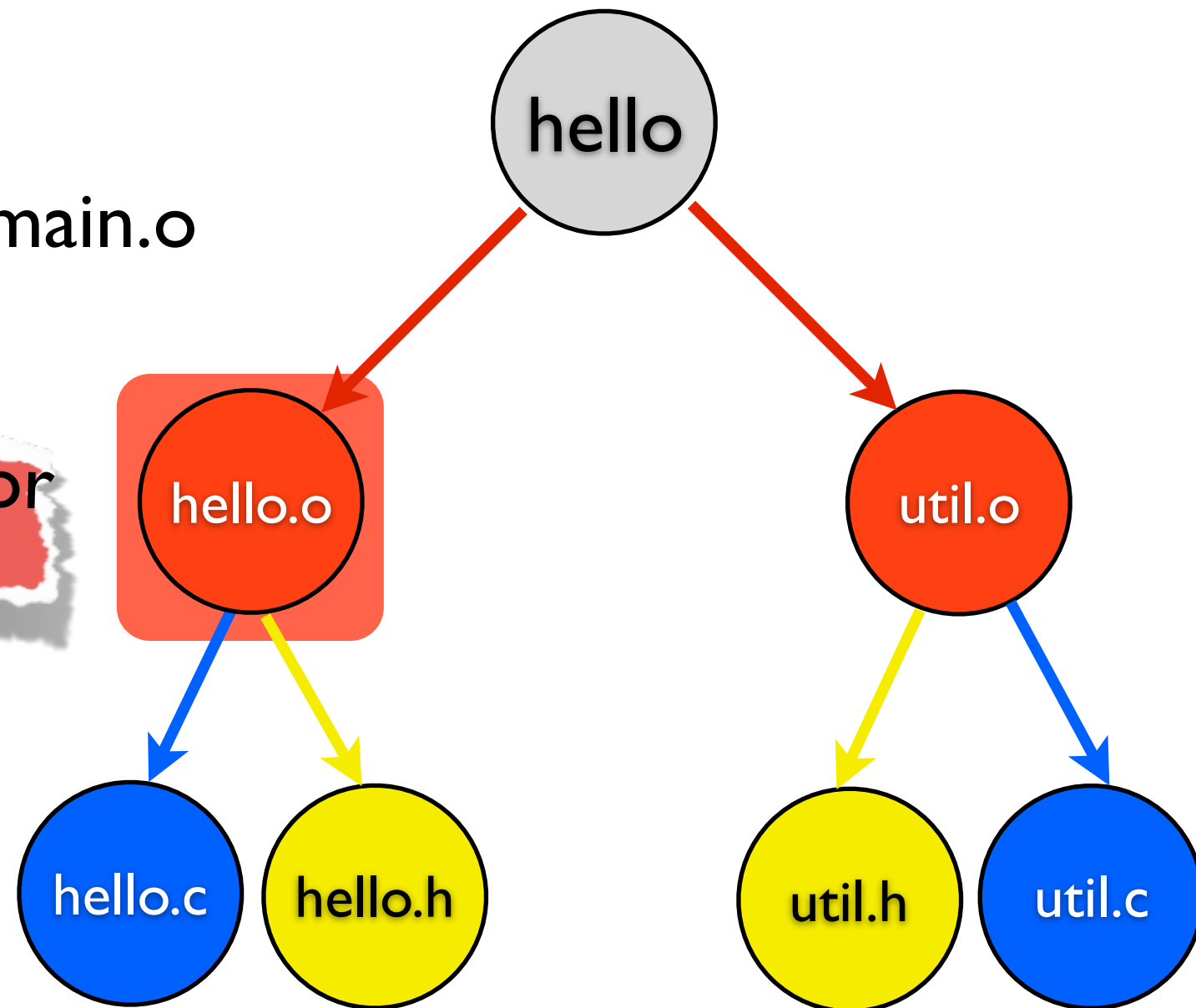
No re-compilation necessary for
util.o!

util.o: util.c util.h

gcc -o util.o -c util.c

main.o: main.c hello.h util.h

gcc -o main.o -c main.c





An incremental build after changing hello.h

hello: hello.o util.o **main.o**

gcc -o hello hello.o util.o main.o

hello.o: hello.c hello.h

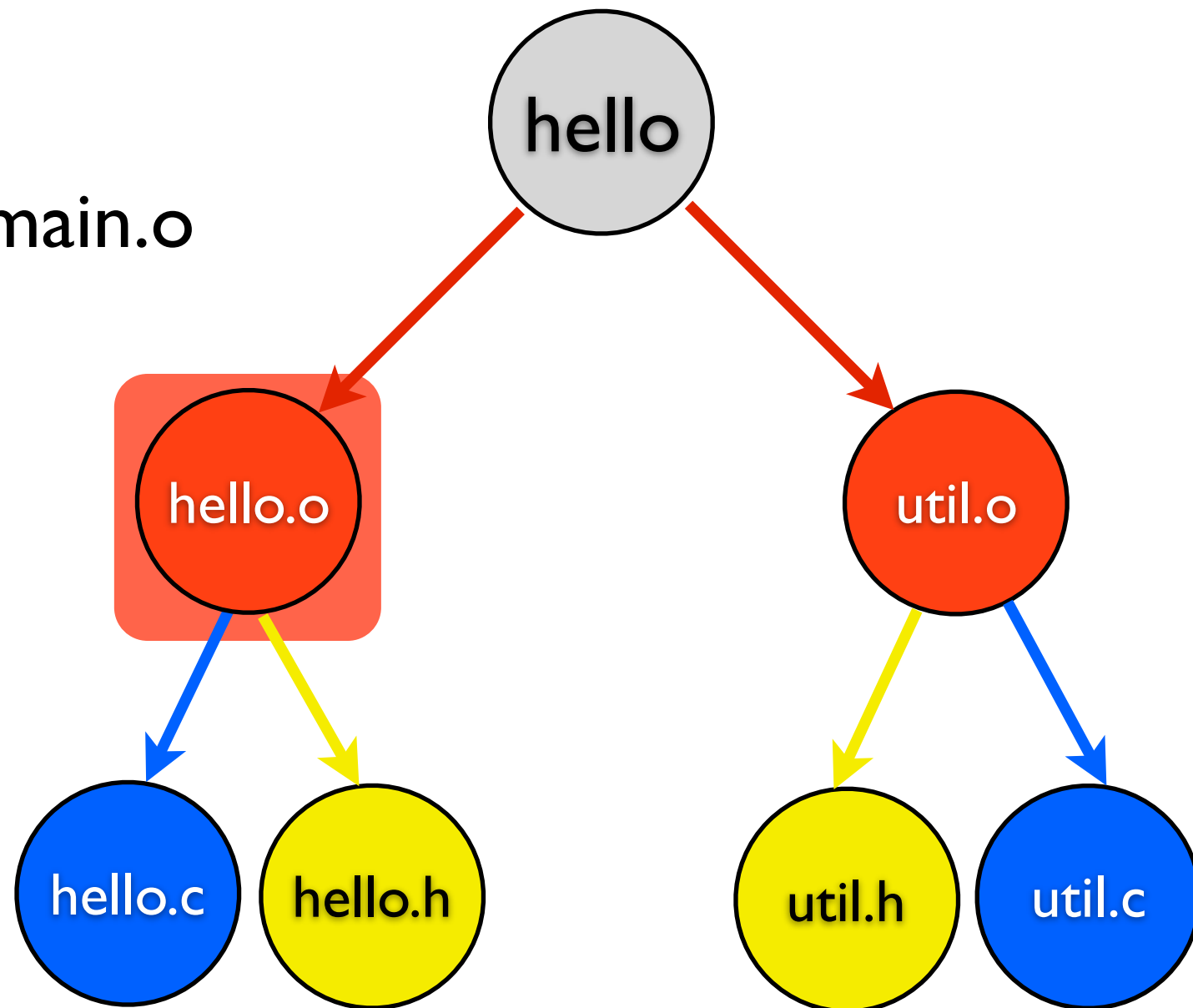
gcc -o hello.o -c hello.c

util.o: util.c util.h

gcc -o util.o -c util.c

main.o: main.c hello.h util.h

gcc -o main.o -c main.c





An incremental build after changing hello.h

hello: hello.o util.o **main.o**

gcc -o hello hello.o util.o main.o

hello.o: hello.c hello.h

gcc -o hello.o -c hello.c

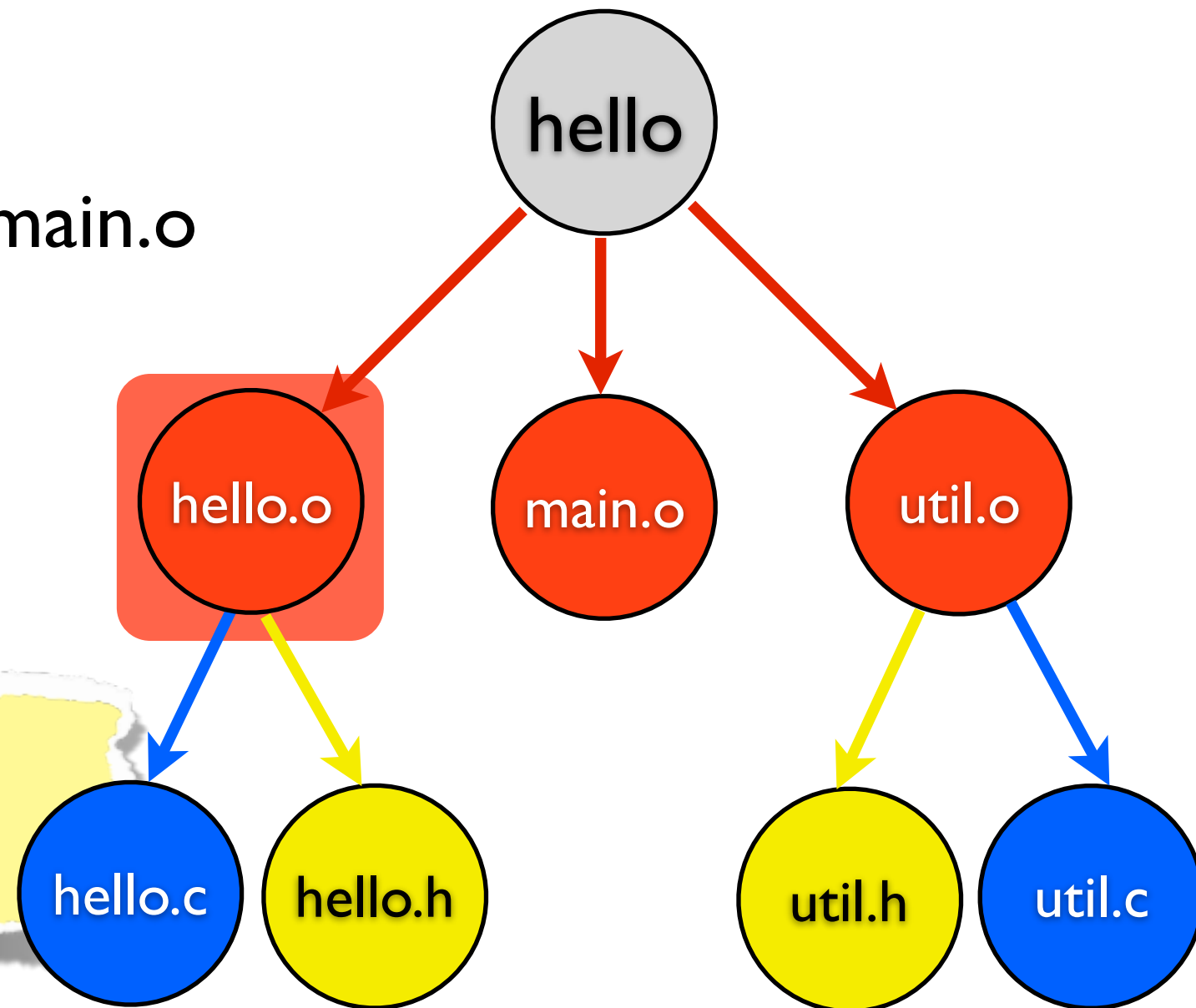
util.o: util.c util.h

does main.o exist? YES

is main.o newer than all its dependencies?

main.o main.c hello.h util.h

gcc -o main.o -c main.c



An incremental build after changing hello.h

hello: hello.o util.o main.o

gcc -o hello hello.o util.o main.o

hello.o: hello.c hello.h

gcc -o hello.o -c hello.c

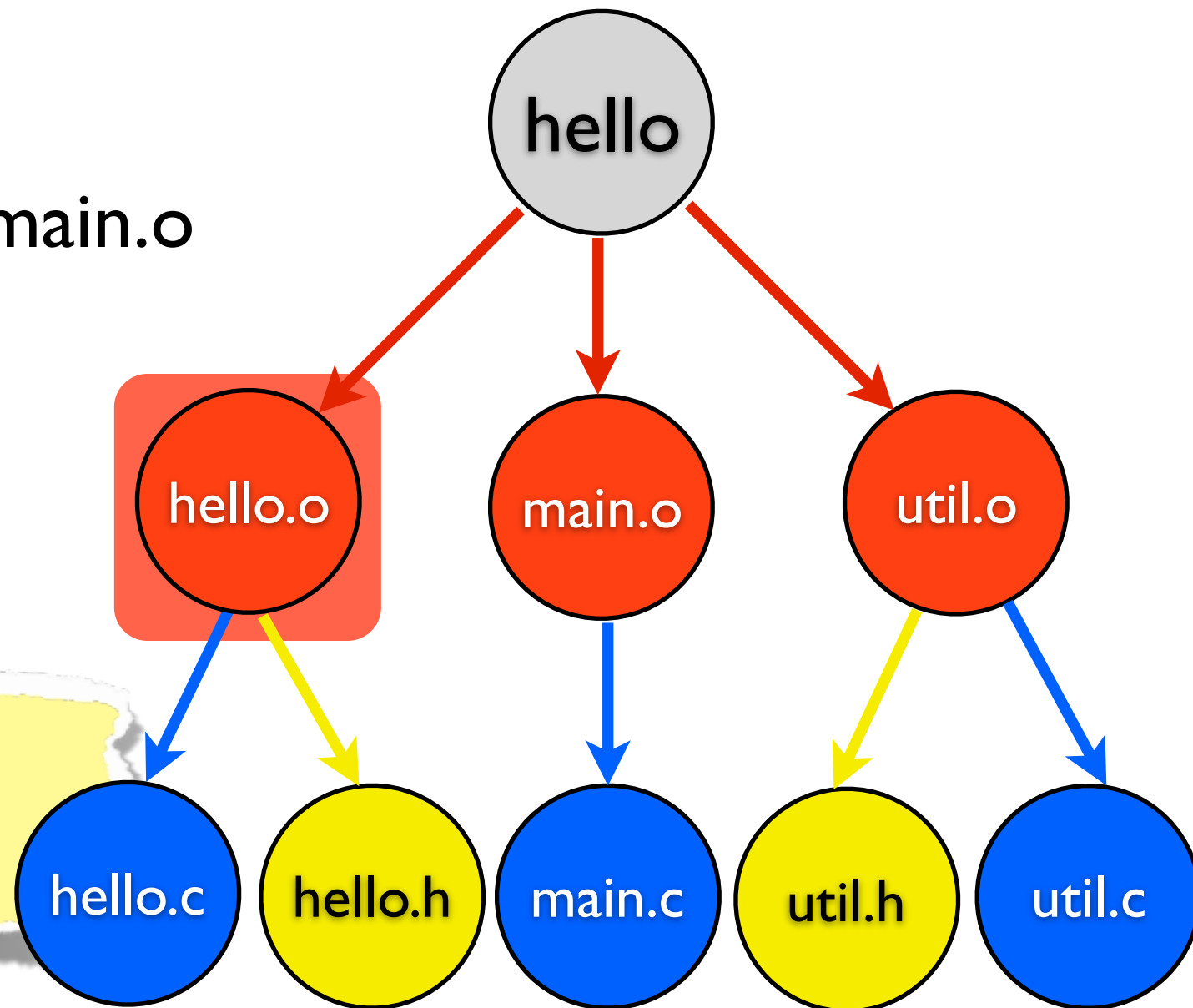
util.o: util.c util.h

does main.o exist? YES

is main.o newer than all its dependencies?

main.o: main.c hello.h util.h

does main.c exist? YES



An incremental build after changing hello.h

hello: hello.o util.o main.o

gcc -o hello hello.o util.o main.o

hello.o: hello.c hello.h

gcc -o hello.o -c hello.c

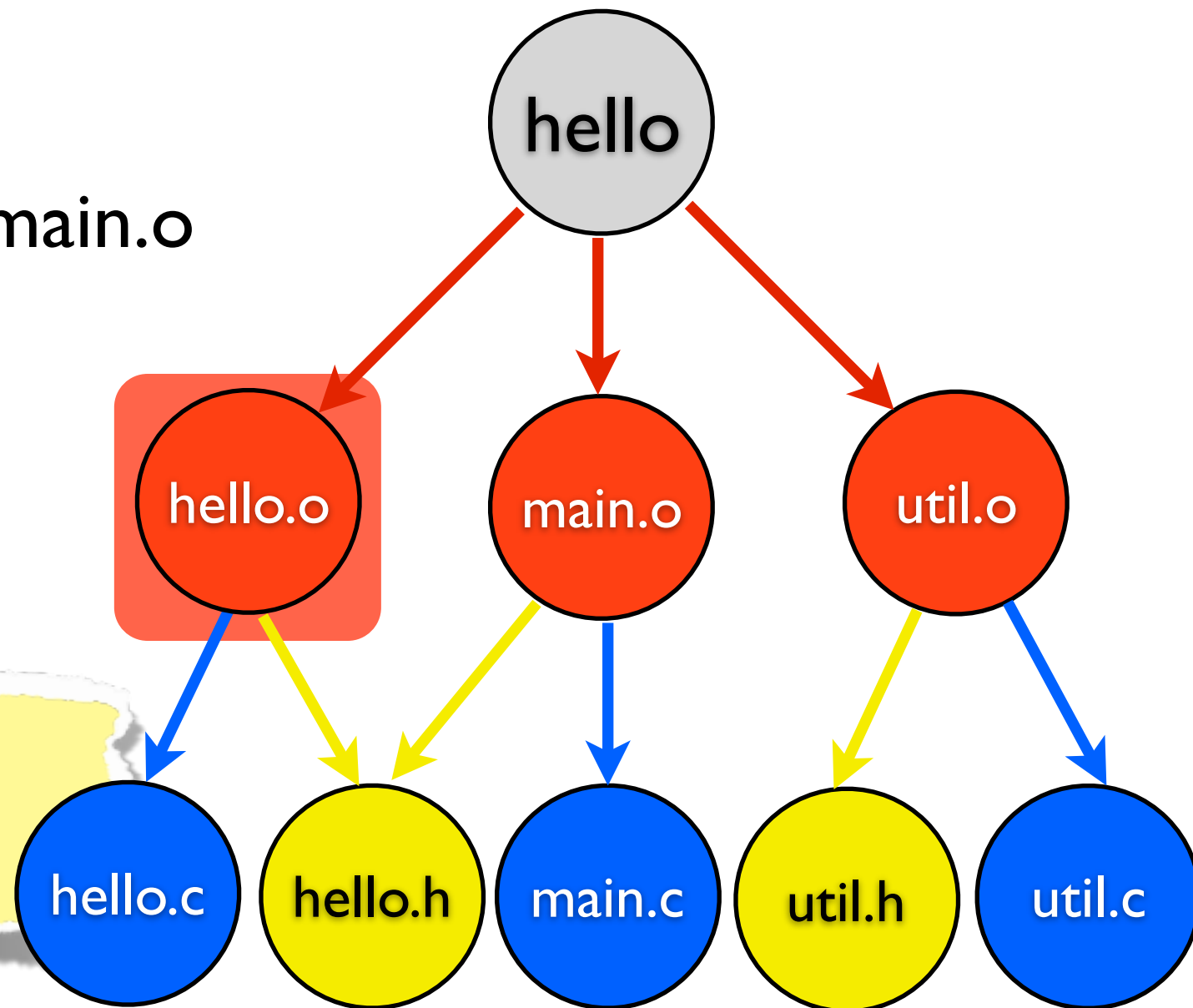
util.o: util.c util.h

does main.o exist? YES

is main.o newer than all its dependencies?

main.o: main.c hello.h util.h

does hello.h exist? YES



An incremental build after changing hello.h

hello: hello.o util.o main.o

gcc -o hello hello.o util.o main.o

hello.o: hello.c hello.h

gcc -o hello.o -c hello.c

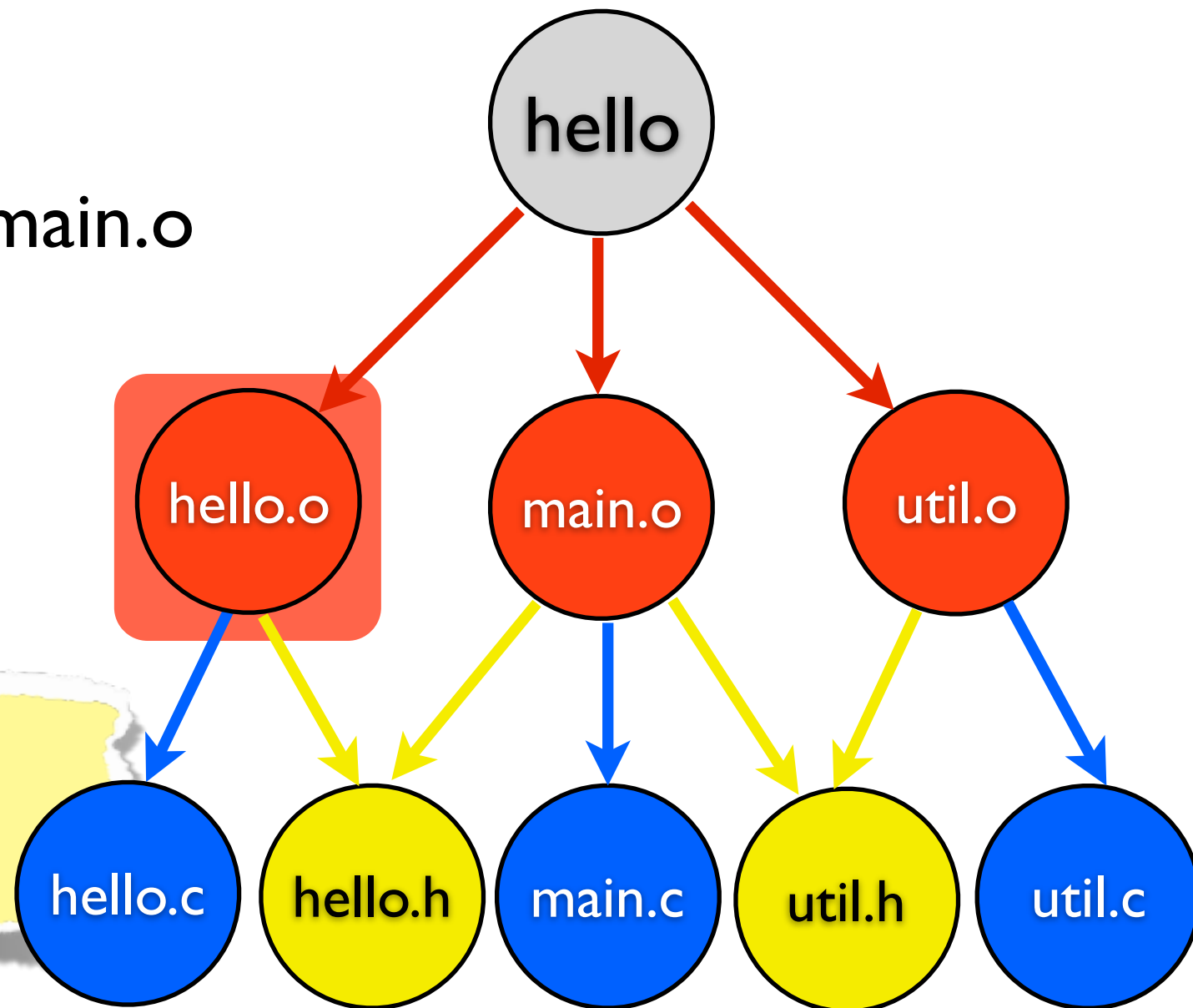
util.o: util.c util.h

does main.o exist? YES

is main.o newer than all its dependencies?

main.o: main.c hello.h util.h

does util.h exist? YES



An incremental build after changing hello.h

hello: hello.o util.o main.o

gcc -o hello hello.o util.o main.o

hello.o: hello.c hello.h

gcc -o hello.o -c hello.c

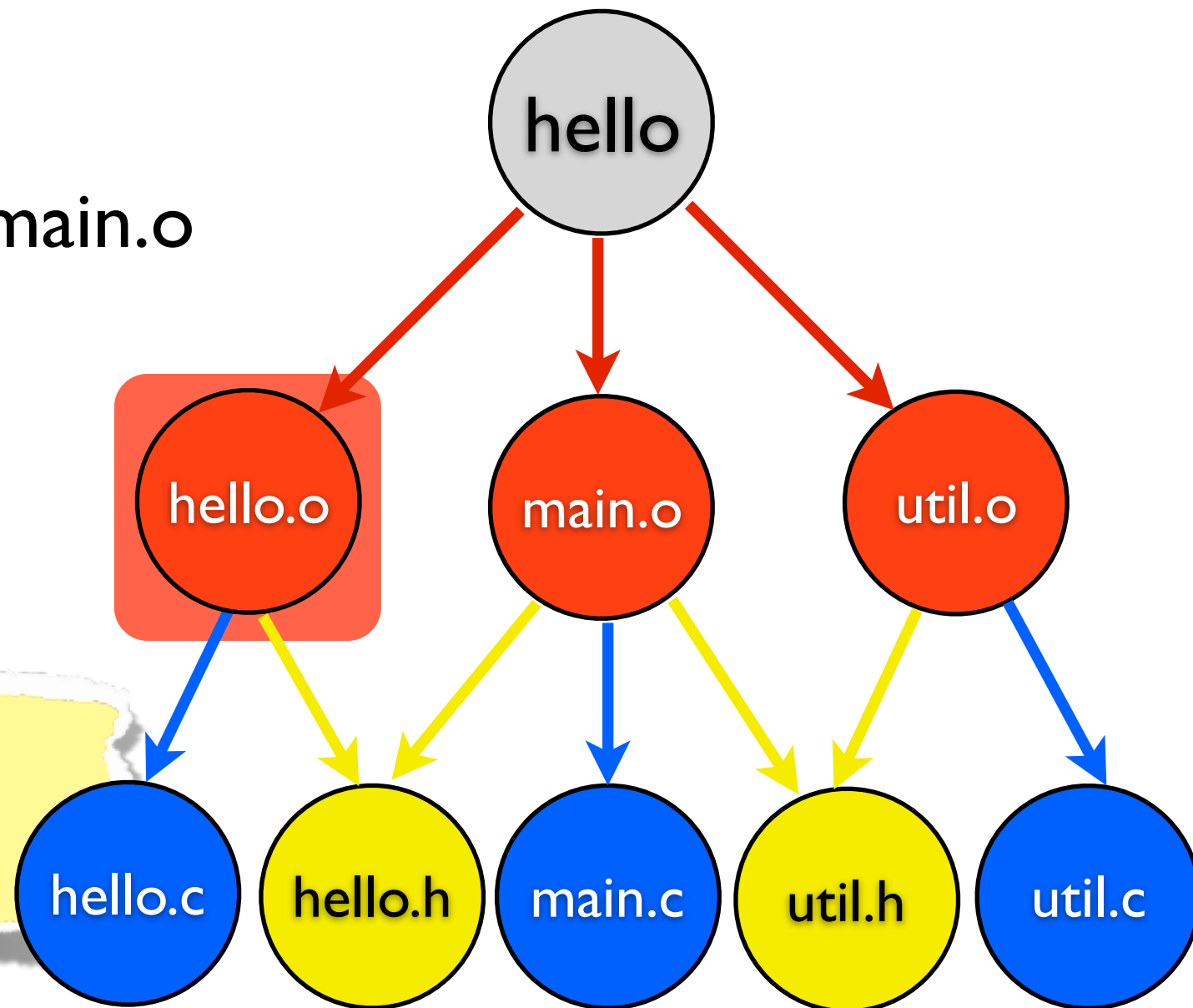
util.o: util.c util.h

does main.o exist? YES

is main.o newer than all its dependencies? **NO**

main.o: main.c hello.h util.h

gcc -o main.o -c main.c



An incremental build after changing hello.h

hello: hello.o util.o main.o

gcc -o hello hello.o util.o main.o

hello.o: hello.c hello.h

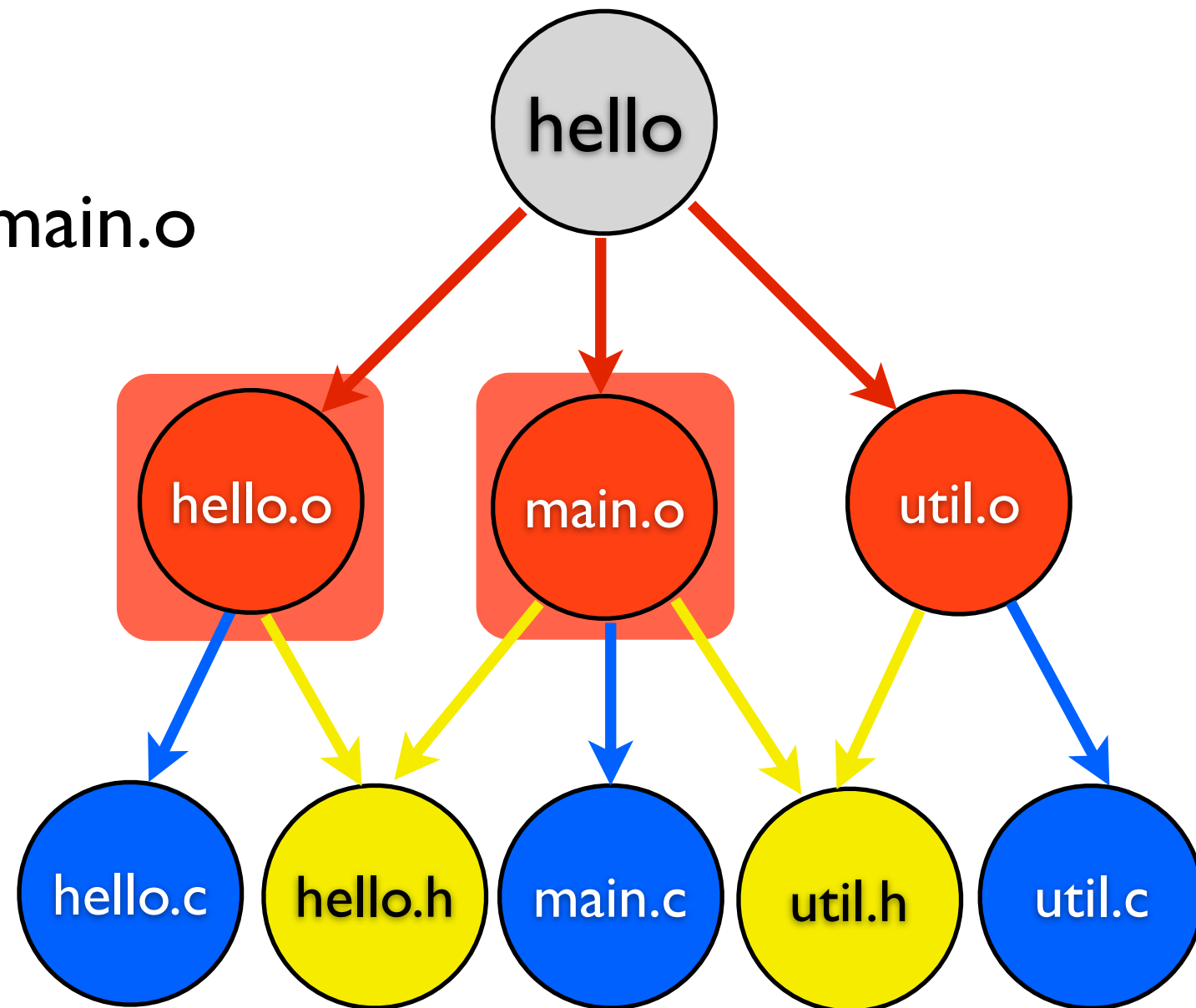
gcc -o hello.o -c hello.c

util.o: util.c util.h

gcc -o util.o -c util.c

main.o: main.c hello.h util.h

gcc -o main.o -c main.c





An incremental build after changing hello.h

hello: hello.o util.o main.o

`gcc -o hello hello.o util.o main.o`

hello.o: hello.c hello.h

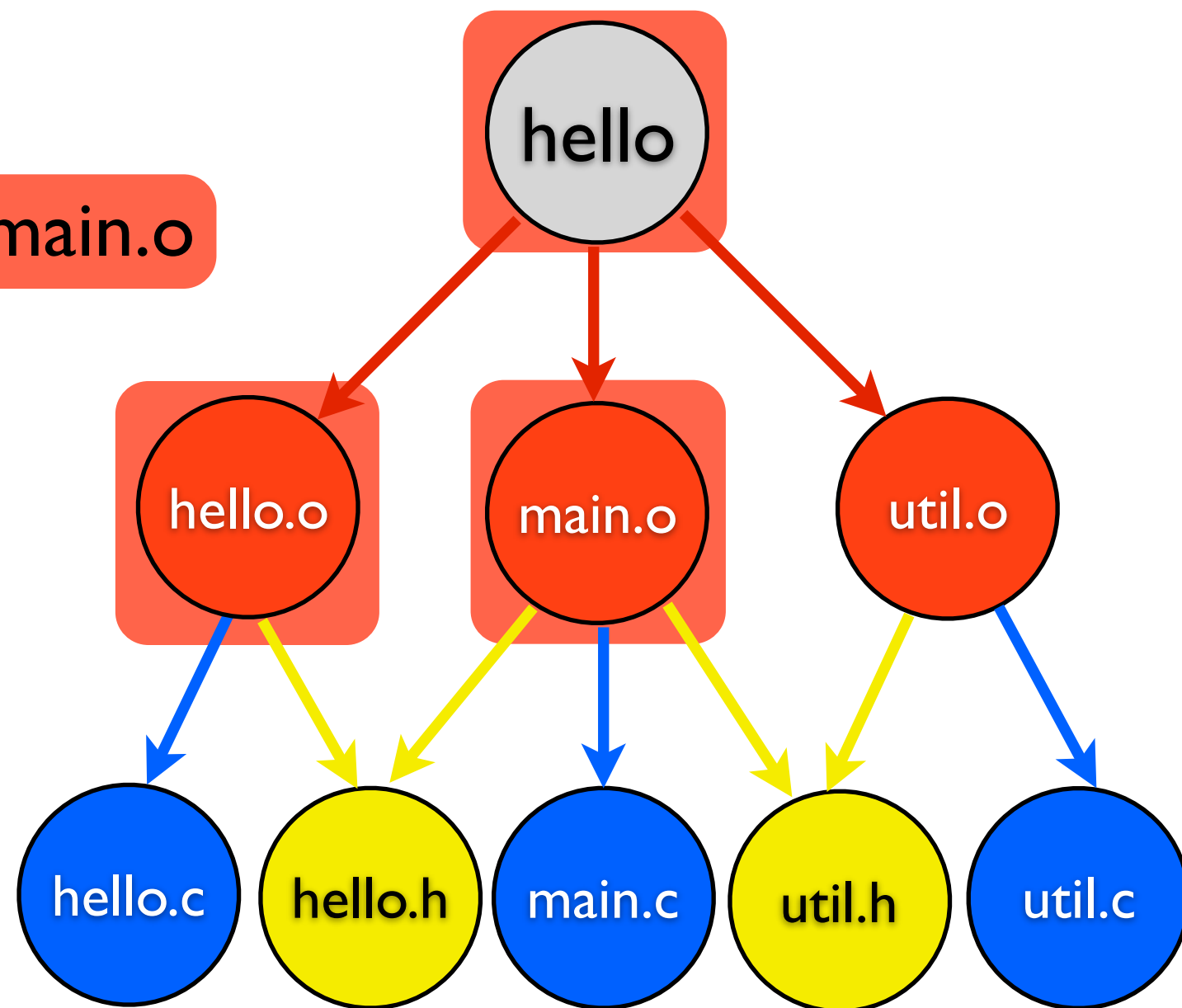
`gcc -o hello.o -c hello.c`

util.o: util.c util.h

`gcc -o util.o -c util.c`

main.o: main.c hello.h util.h

`gcc -o main.o -c main.c`





An incremental build after changing hello.h

hello: hello.o util.o main.o

gcc -o hello hello.o util.o main.o

hello.o: hello.c hello.h

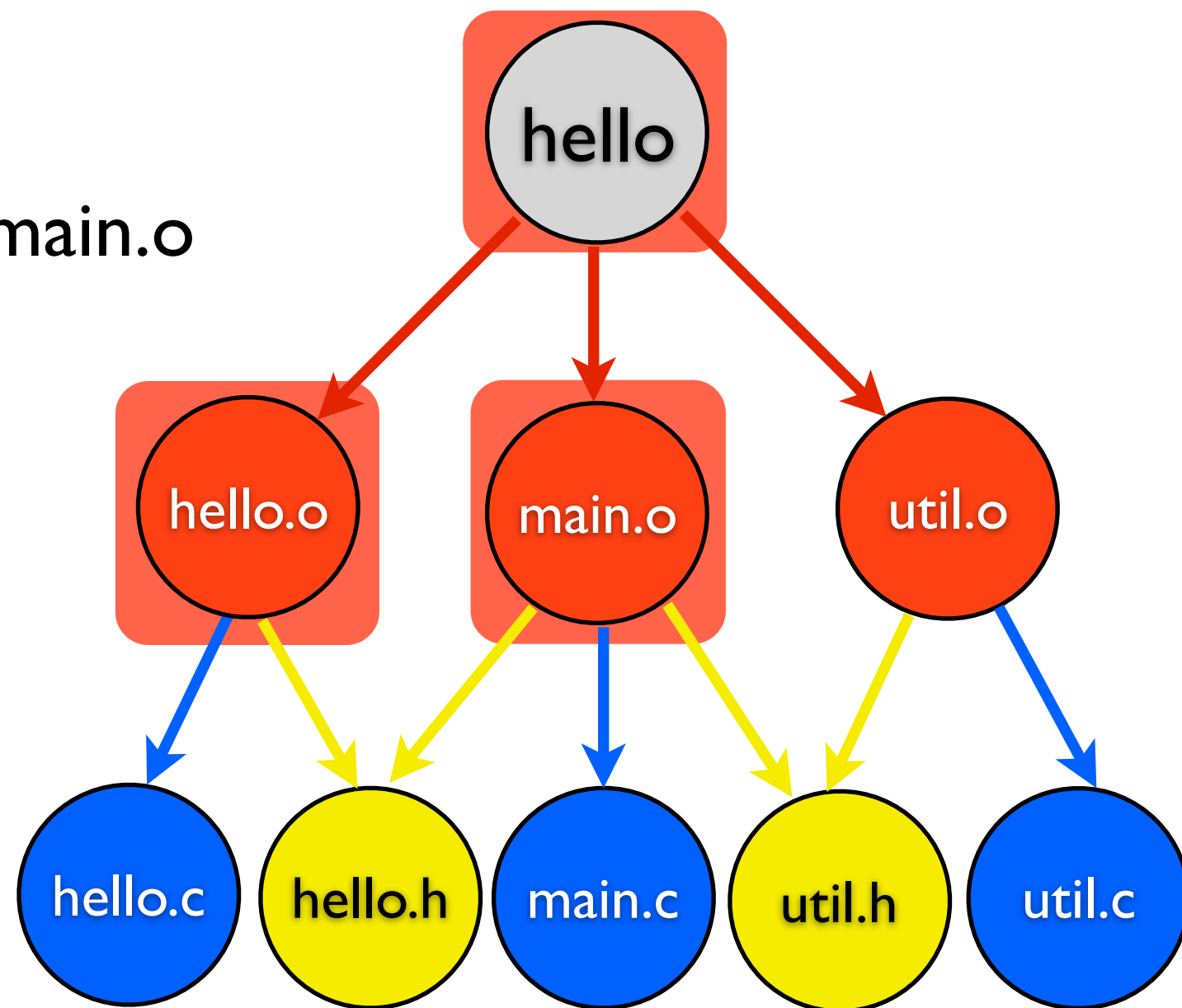
gcc -o hello.o -c hello.c

util.o: util.c util.h

gcc -o util.o -c util.c

main.o: main.c hello.h util.h

gcc -o main.o -c main.c



only **3** build commands executed,
so we won (build) time!

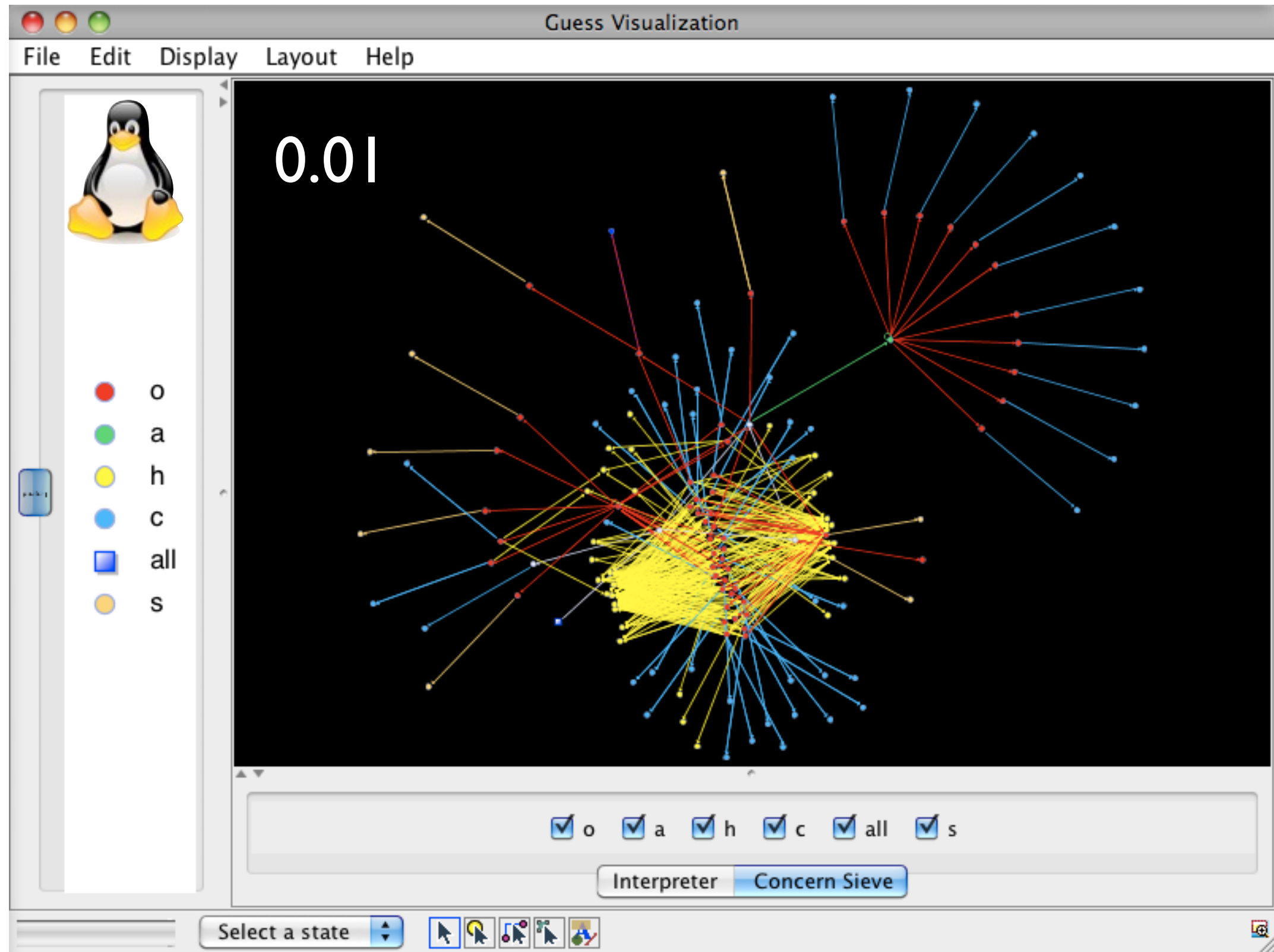


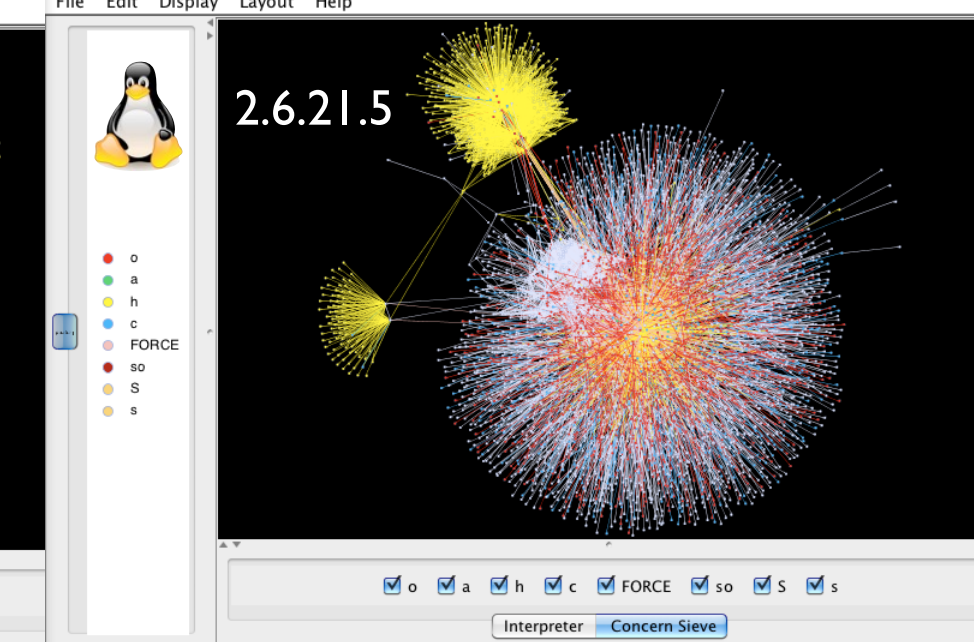
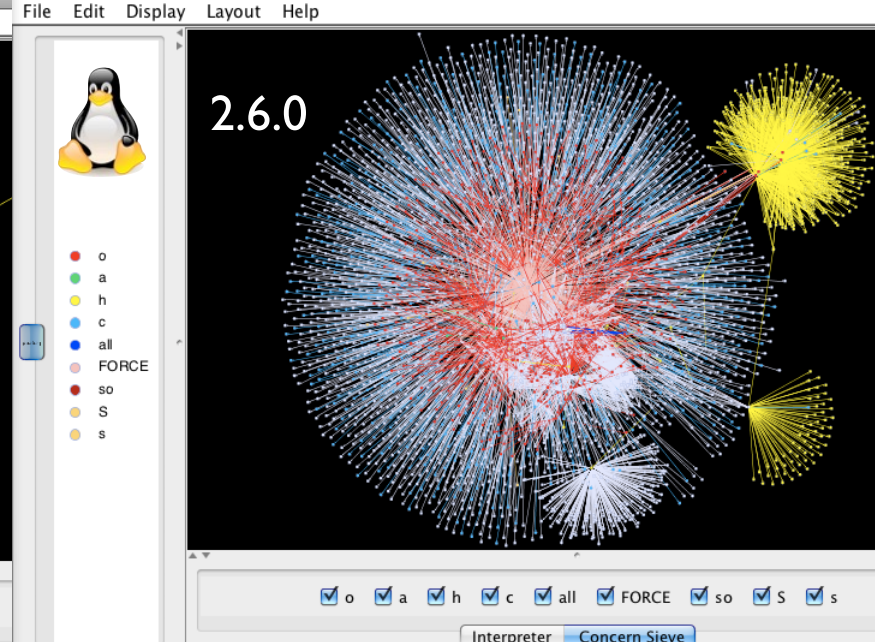
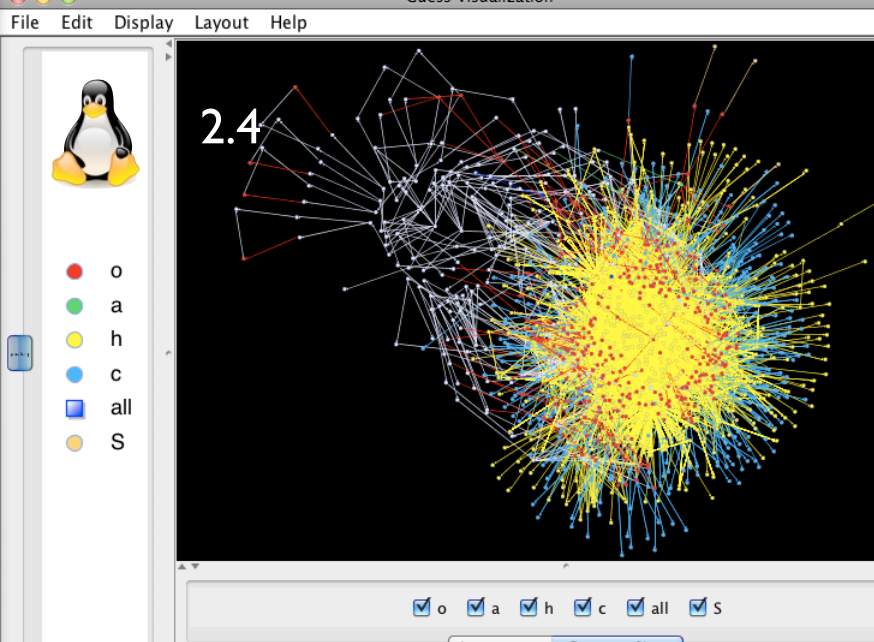
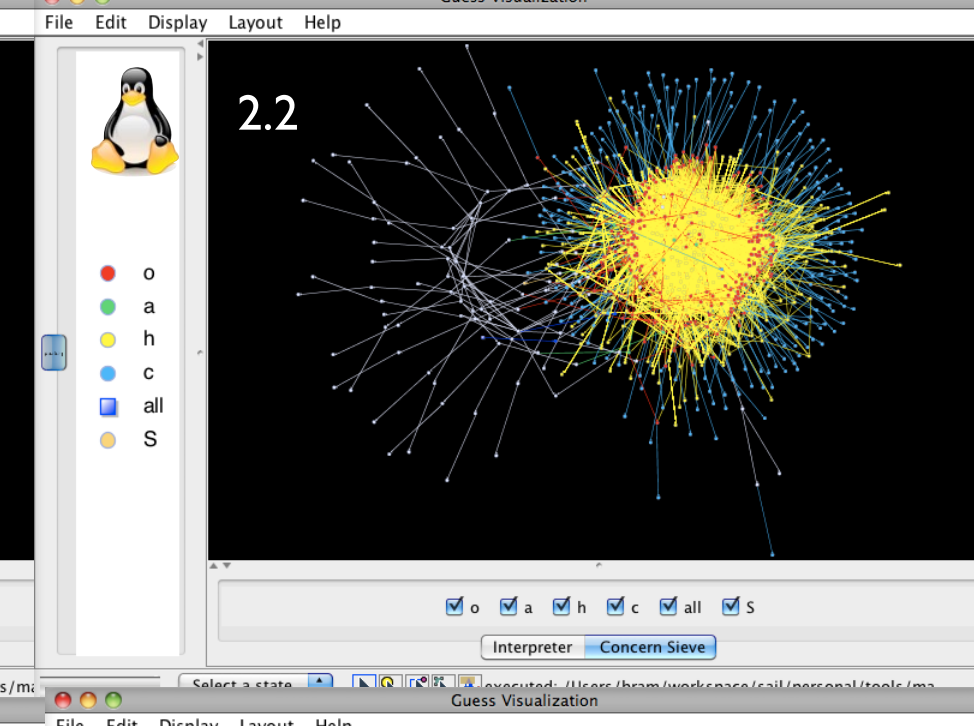
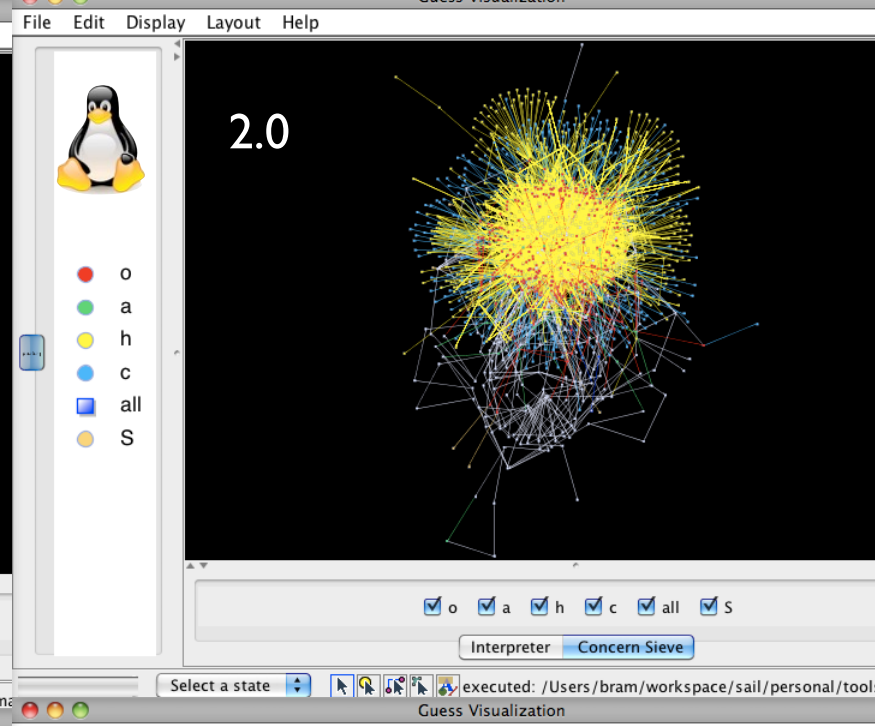
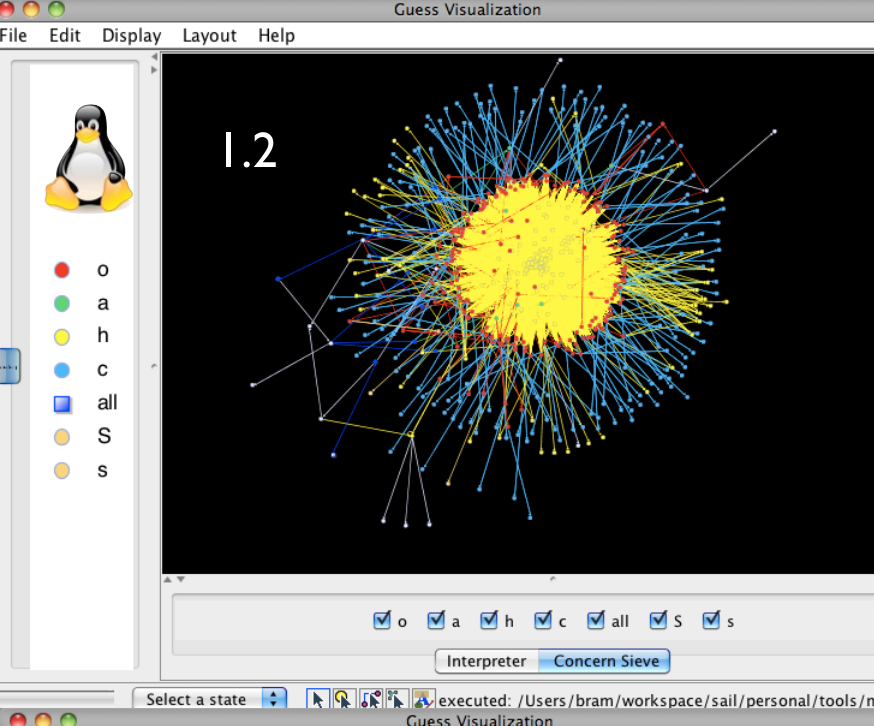
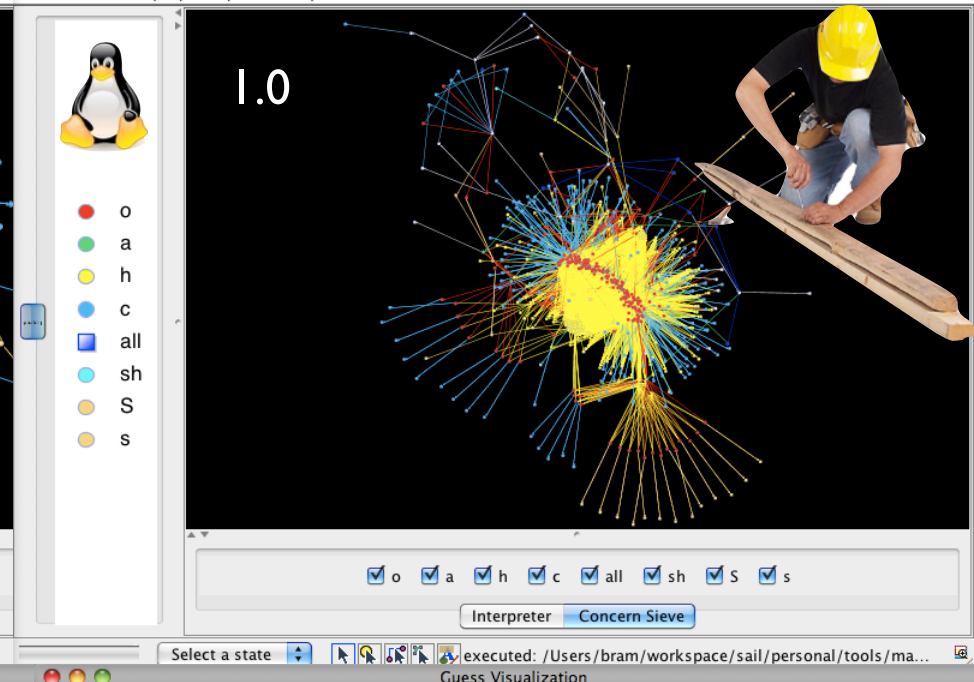
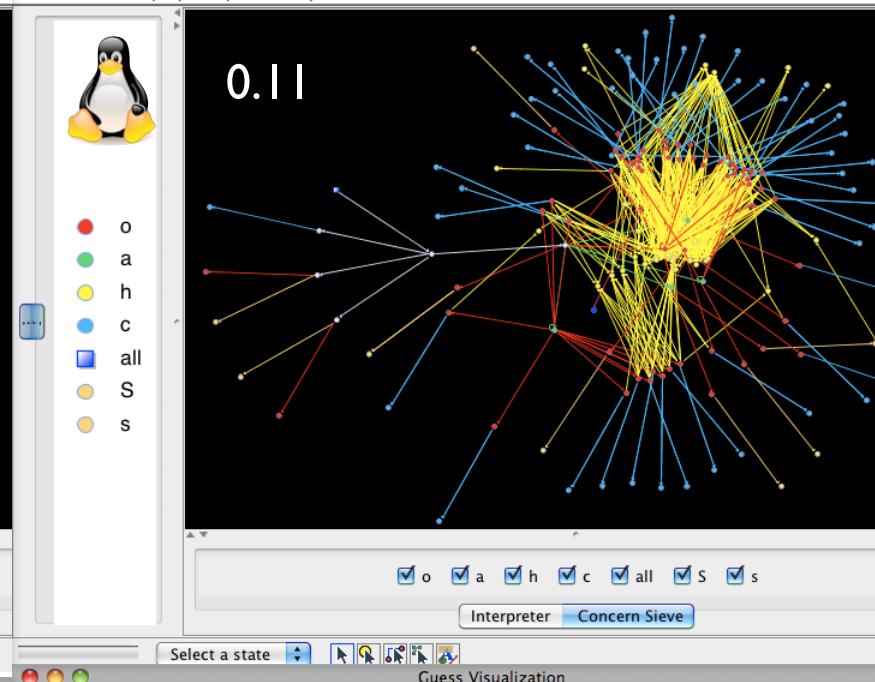
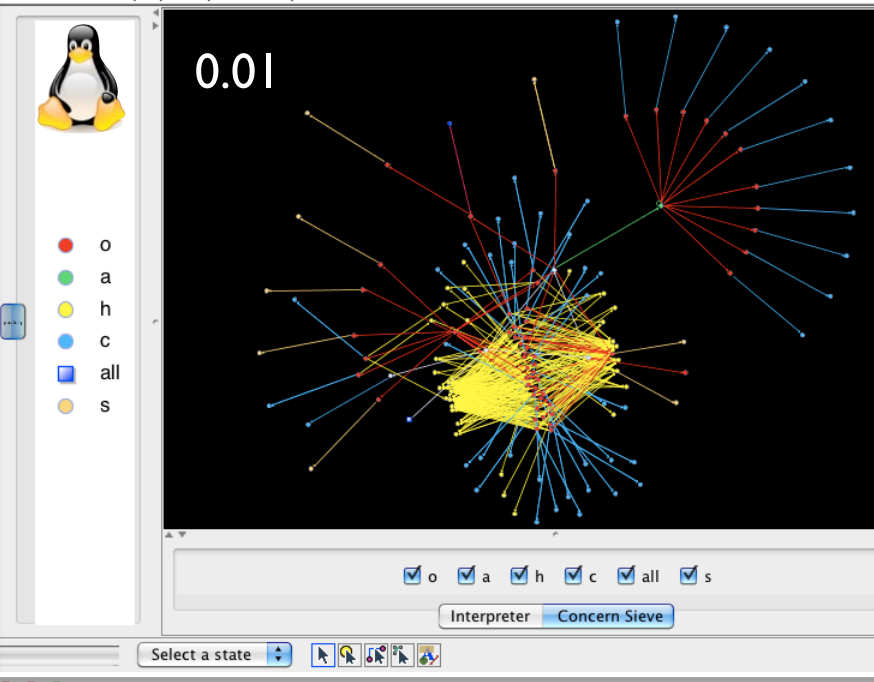
"The Evolution of the Linux Build System" (Adams et al.)

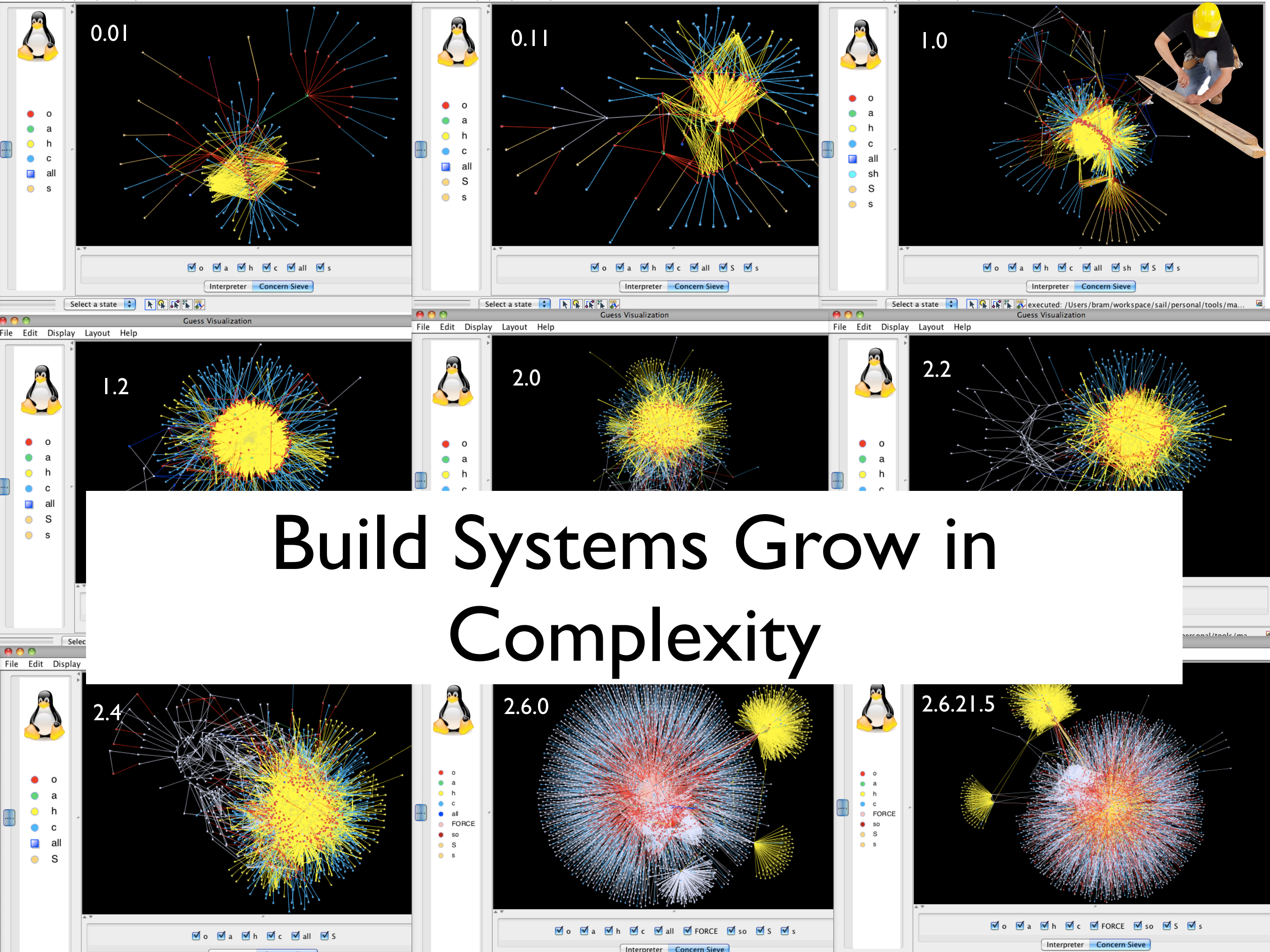




"The Evolution of the Linux Build System" (Adams et al.)







There is Way More than Just GNU Make!

List of build automation software

From Wikipedia, the free encyclopedia

Build automation involves [scripting](#) or automating the process of [compiling](#) computer [source code](#) into [binary code](#). Below is a list of notable tools associated with automating build processes

Contents [\[hide\]](#)

- 1 [Make-based tools](#)
- 2 [Non-Make-based tools](#)
- 3 [Build script generation tools](#)
- 4 [Continuous integration tools](#)
- 5 [Configuration management tools](#)
- 6 [Meta-build tools](#)
- 7 [Other tools](#)
- 8 [Comparison of build automation software](#)
- 9 [References](#)
- 10 [External links](#)

Make-based tools [\[edit \]](#)

- [GNU make](#), a widely used make implementation with a large set of extensions
- [make](#), a classic Unix build tool
- [mk](#), developed originally for [Version 10 Unix](#) and [Plan 9](#), and ported to Unix as part of [plan9port](#)
- [MPW Make](#), developed for the [classic Mac OS](#) and similar to but not compatible with Unix make; the modern [macOS](#) (OS X) comes with both GNU make and BSD make; available as part of Macintosh Programmer's Workshop as a free, unsupported download from Apple
- [nmake](#)
- [PVCS-make](#), basically follows the concept of [make](#) but with a noticeable set of unique syntax features^[1]

There is Way More than Just GNU Make!

List of build automation software

From Wikipedia, the free encyclopedia

Build automation involves [scripting](#) or automating the process of [compiling](#) computer [source code](#) into [binary code](#). Below is a list of notable tools associated with automating build processes

Contents [\[hide\]](#)

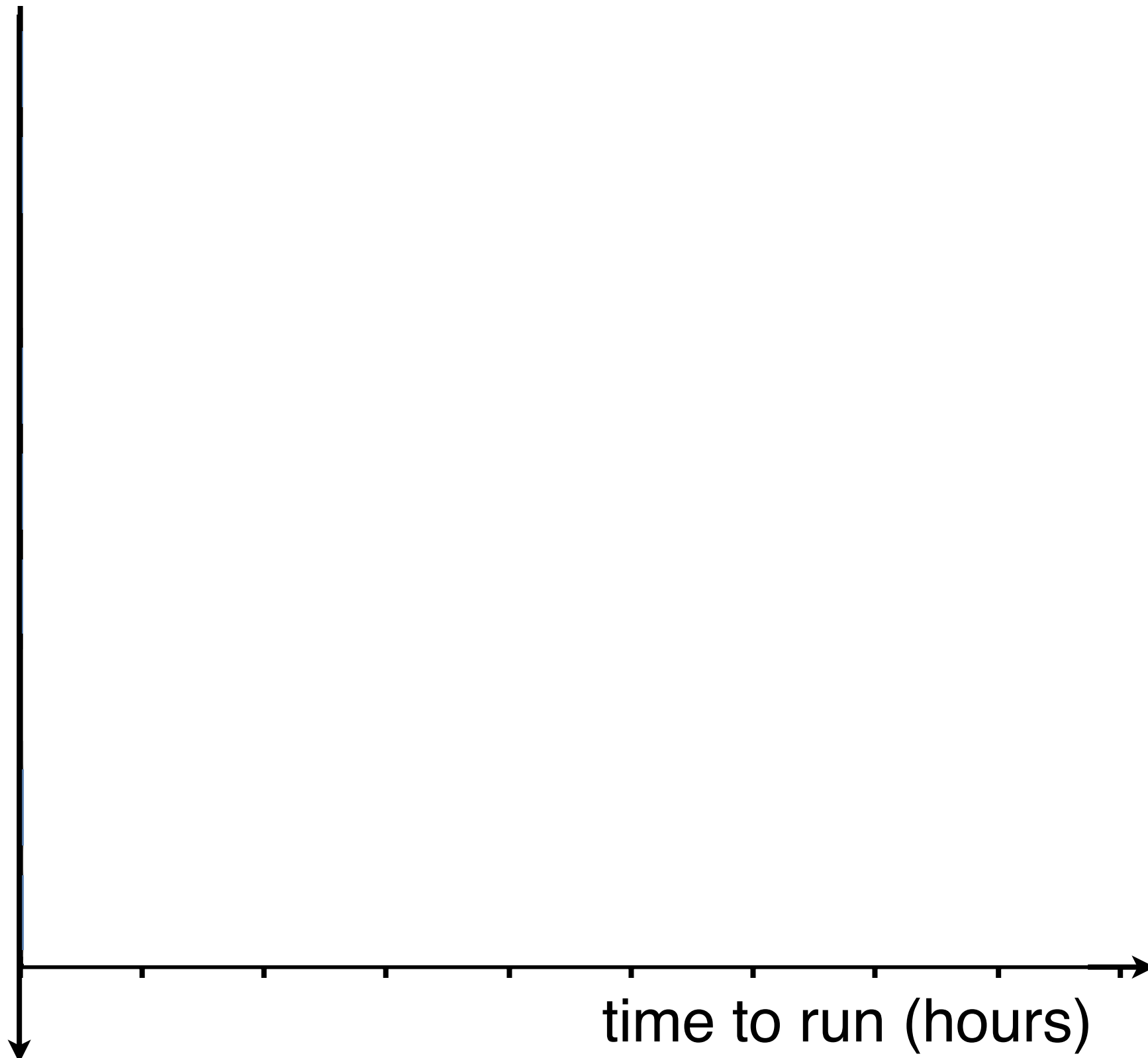
- 1 [Make-based tools](#)
- 2 [Non-Make-based tools](#)
- 3 [Build script generation tools](#)
- 4 [Continuous integration tools](#)
- 5 [Configuration management tools](#)
- 6 [Meta-build tools](#)
- 7 [Other tools](#)
- 8 [Comparison of build automation software](#)
- 9 [References](#)
- 10 [External links](#)

6 Make-based tools
28 Non-Make-based tools
10 build script generation tools
16 CI tools
9 IaC tools

Make-based tools [\[edit\]](#)

- [GNU make](#), a widely used make implementation with a large set of extensions
- [make](#), a classic Unix build tool
- [mk](#), developed originally for [Version 10 Unix](#) and [Plan 9](#), and ported to Unix as part of [plan9port](#)
- [MPW Make](#), developed for the [classic Mac OS](#) and similar to but not compatible with Unix make; the modern [macOS](#) (OS X) comes with both GNU make and BSD make; available as part of Macintosh Programmer's Workshop as a free, unsupported download from Apple
- [nmake](#)
- [PVCS-make](#), basically follows the concept of [make](#) but with a noticeable set of unique syntax features^[1]

Why do (CI) Builds Take so Long?



Why do (CI) Builds Take so Long?

local developer build

time to run (hours)



Why do (CI) Builds Take so Long?

incremental compilation
(selection of) unit tests

local developer build

time to run (hours)



Why do (CI) Builds Take so Long?

incremental compilation
(selection of) unit tests

local developer build

CI build of merged change

time to run (hours)



Why do (CI) Builds Take so Long?

incremental compilation
(selection of) unit tests

local developer build

full compilation

all unit tests

integration tests

CI build of merged change

time to run (hours)



Why do (CI) Builds Take so Long?

incremental compilation
(selection of) unit tests

local developer build

full compilation

all unit tests

integration tests

CI build of merged change

closer to release

time to run (hours)



Why do (CI) Builds Take so Long?

incremental compilation
(selection of) unit tests

local developer build

full compilation

all unit tests

integration tests

CI build of merged change

performance tests

user acceptance/system tests

closer to release

time to run (hours)





Even worse ...



Even worse ...



build machinery
runs on each
commit!



Even worse ...



build machinery
runs on each
commit!



different feature
configurations

Even worse ...



build machinery
runs on each
commit!



different feature
configurations

Even worse ...

not just build and
test, also code
quality builds,
nightly builds, etc.



build machinery
runs on each
commit!



different feature
configurations

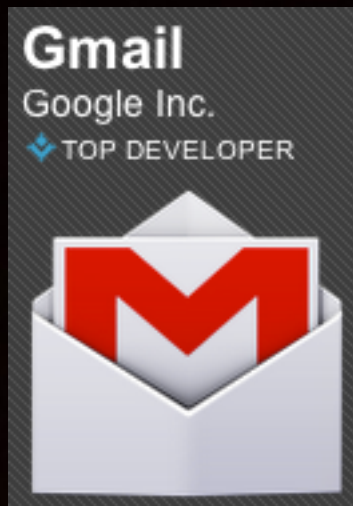
Even worse ...

not just build and
test, also code
quality builds,
nightly builds, etc.

not all builds
succeed

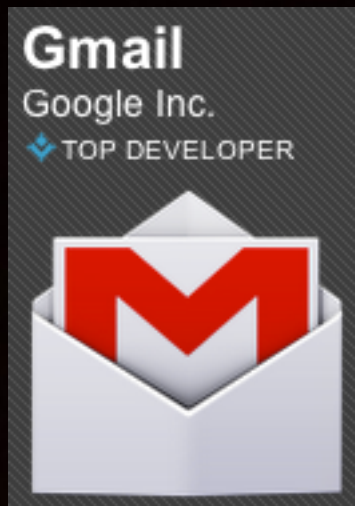


What Features should we Test First?

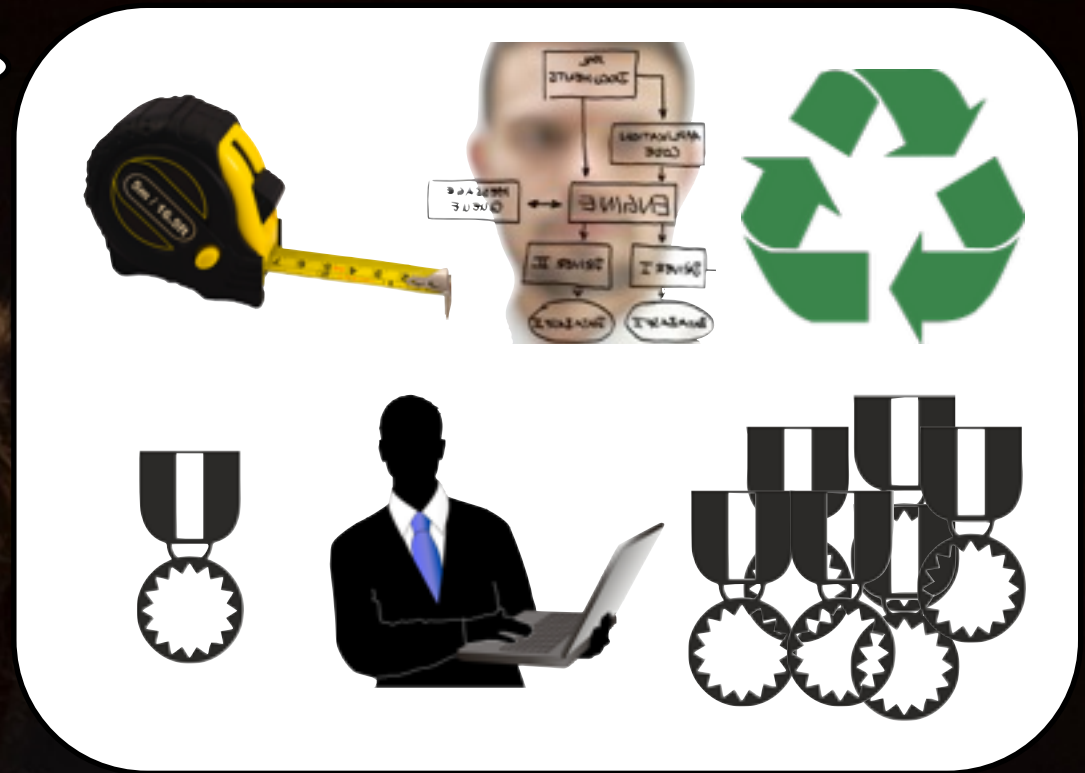


v5.2

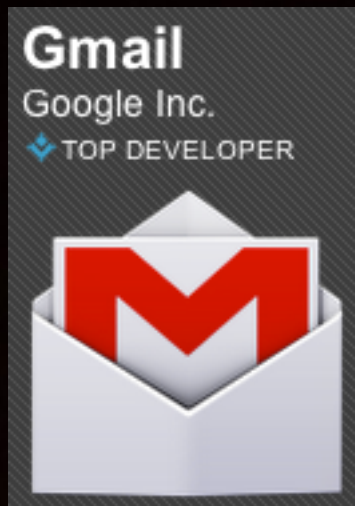
What Features should we Test First?



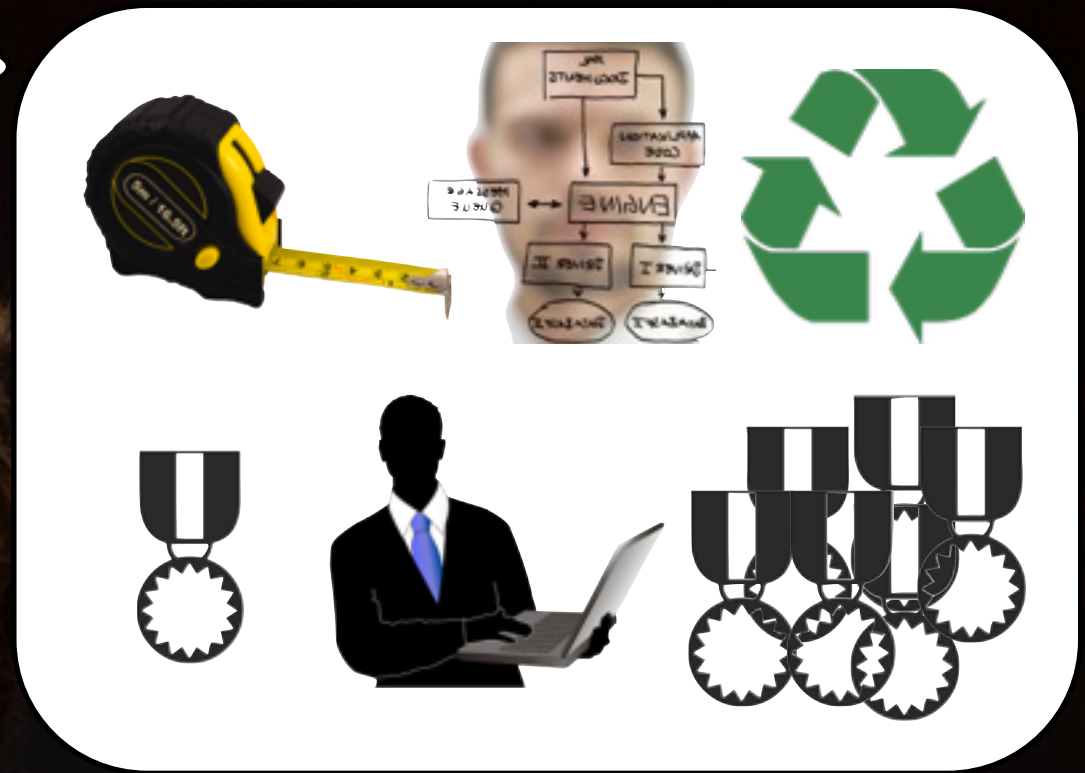
v5.2



What Features should we Test First?



v5.2



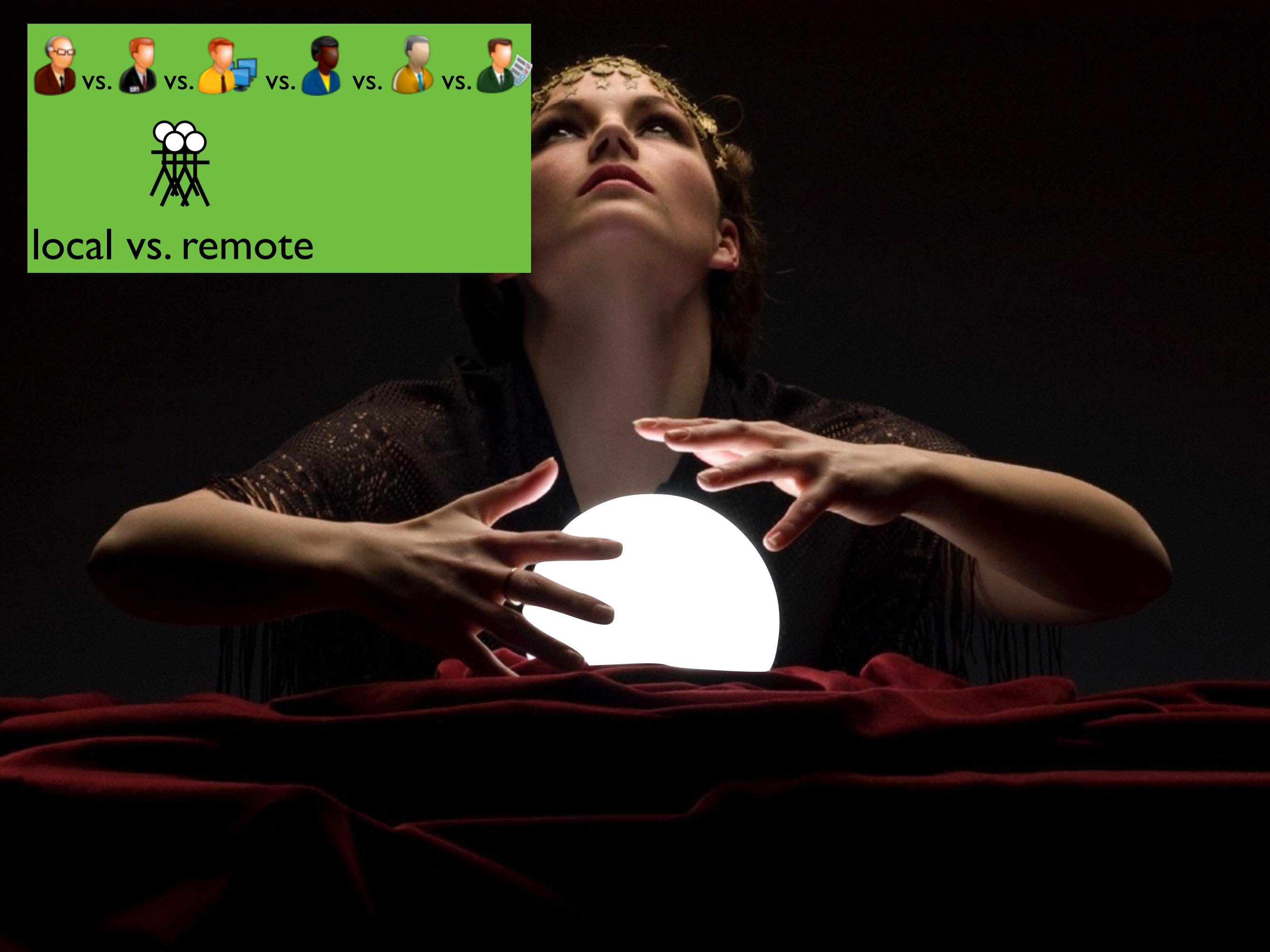
What Features should we Test First?



👤 vs. 👤 vs. 👤 vs. 👤 vs. 👤 vs. 👤

📹

local vs. remote



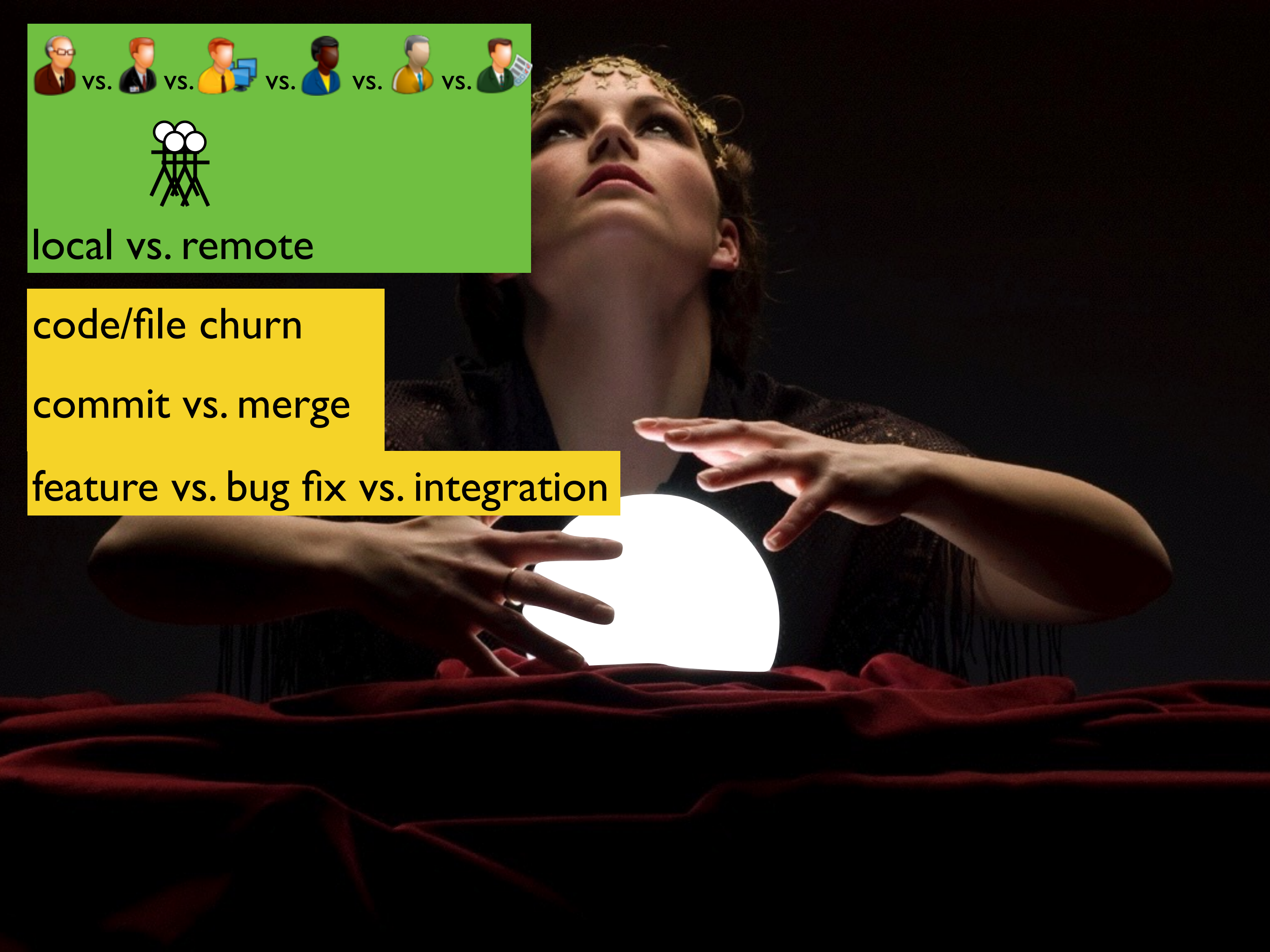


local vs. remote


code/file churn

commit vs. merge

feature vs. bug fix vs. integration



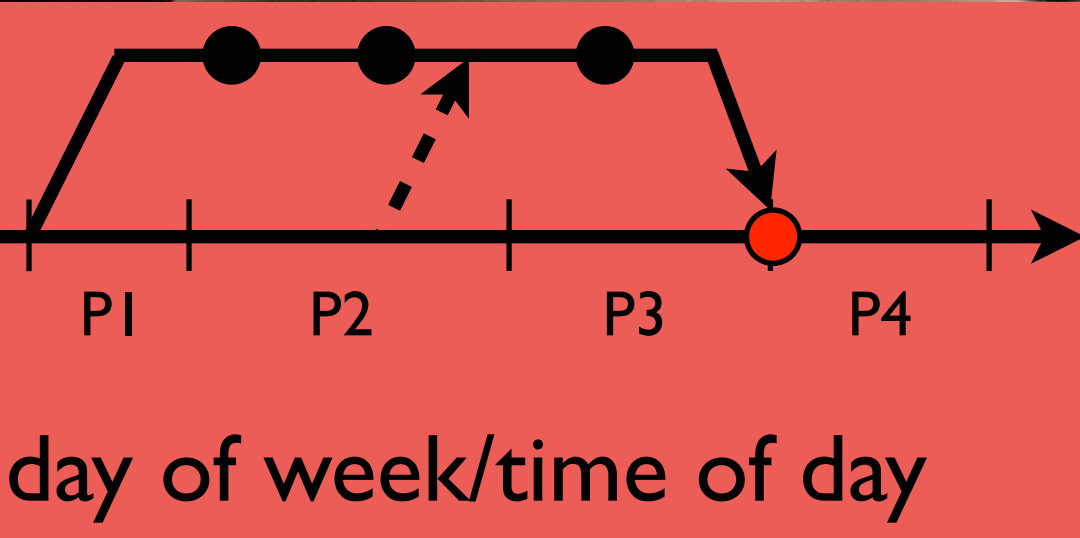
VS. VS. VS. VS. VS. VS.




local vs. remote

code/file churn
commit vs. merge

feature vs. bug fix vs. integration

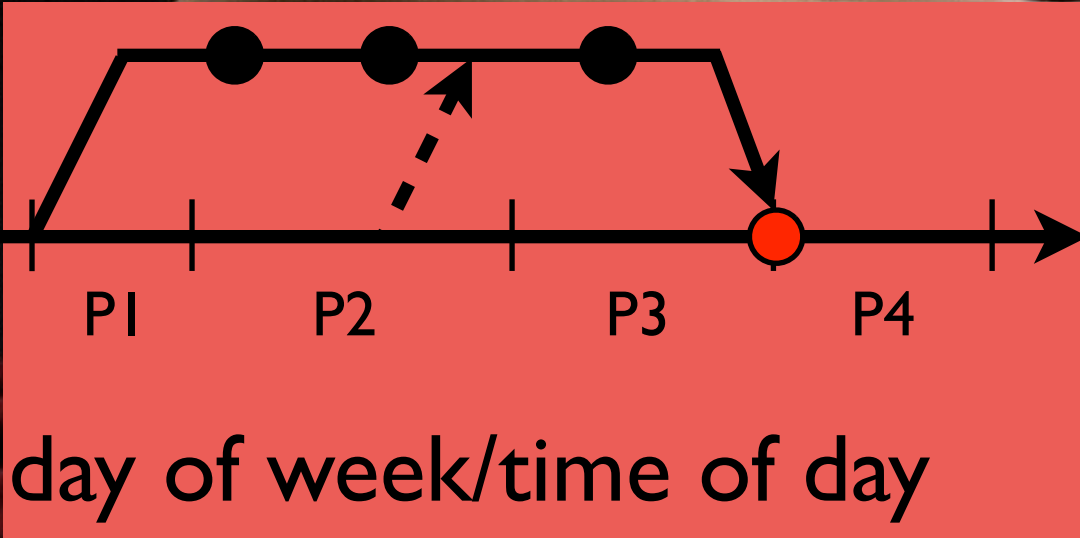


VS. VS. VS. VS. VS. VS.

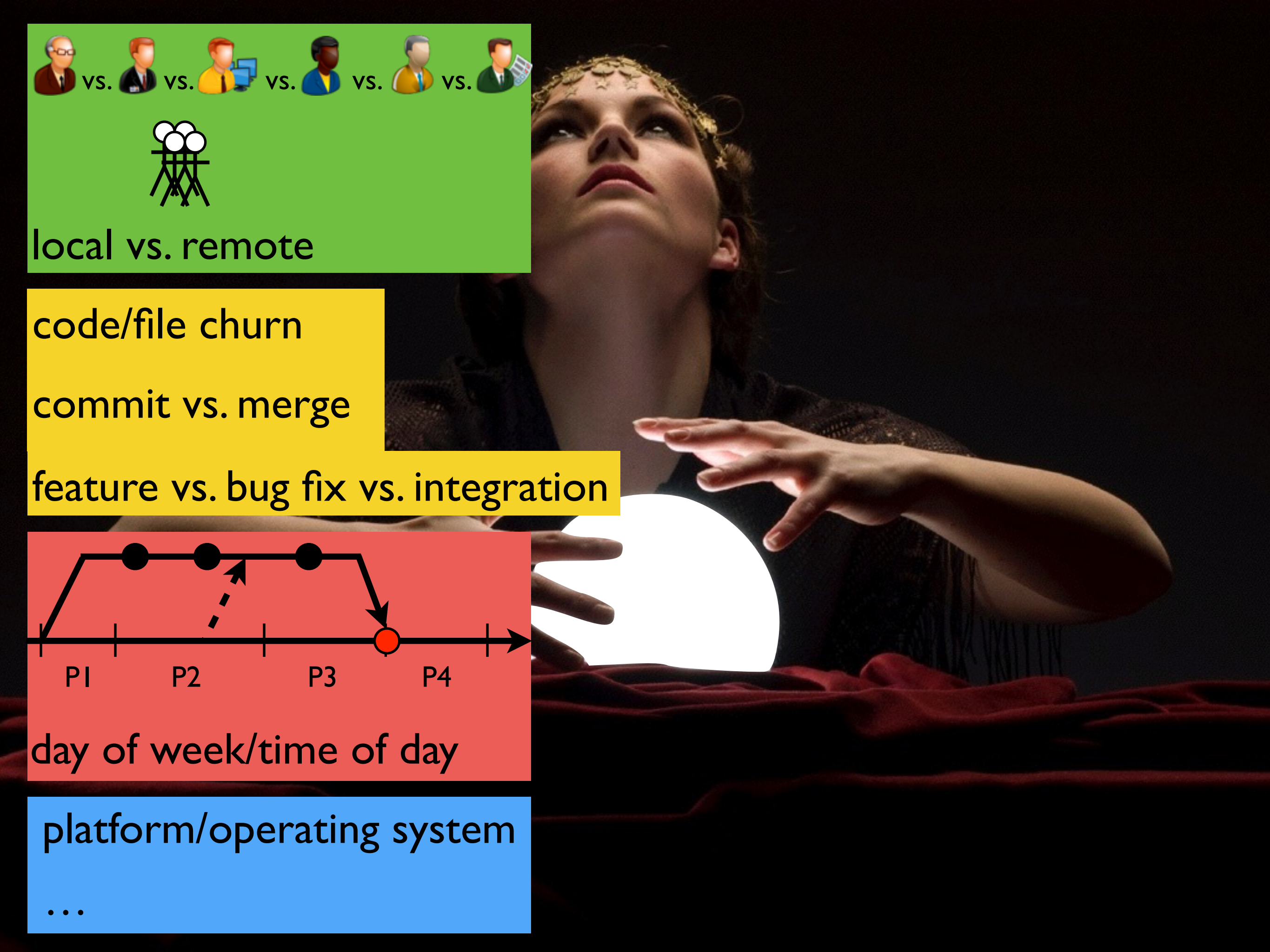


local vs. remote


code/file churn
commit vs. merge
feature vs. bug fix vs. integration



platform/operating system
...



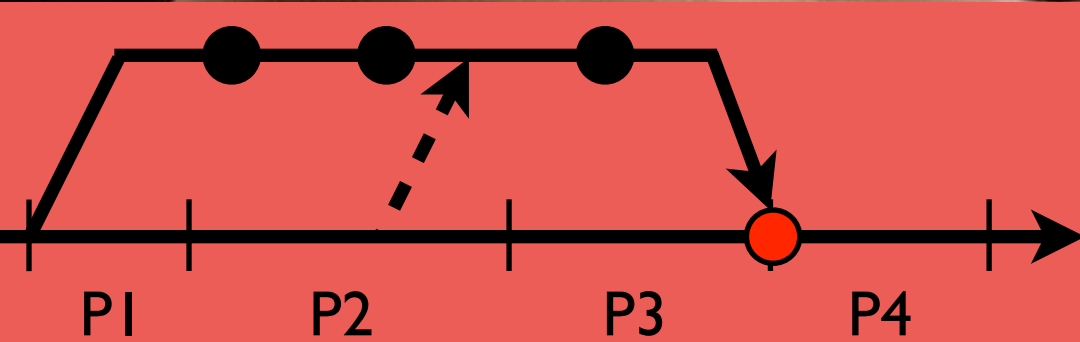
VS. VS. VS. VS. VS. VS.



local vs. remote

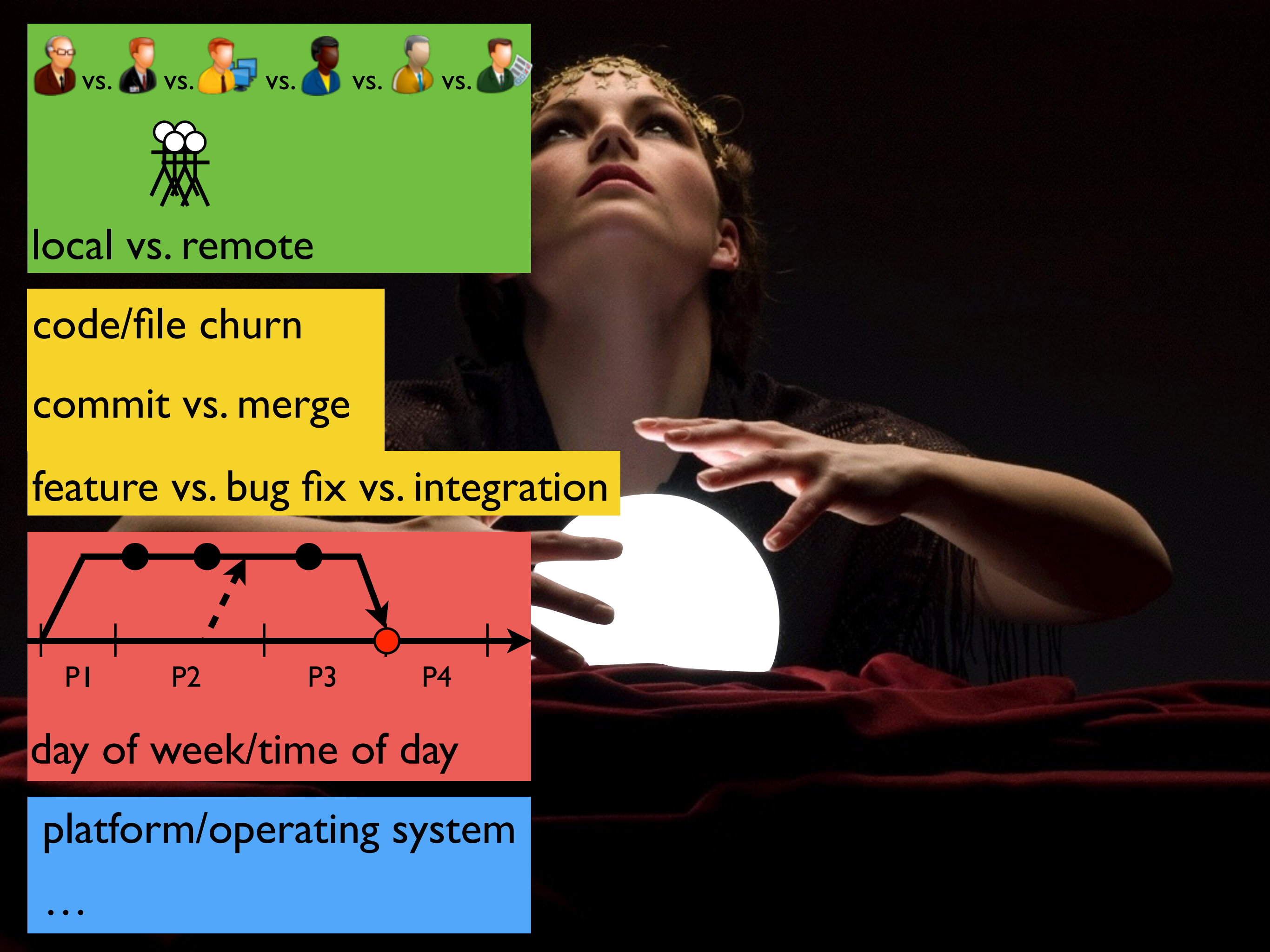
code/file churn
commit vs. merge

feature vs. bug fix vs. integration




day of week/time of day

platform/operating system
...



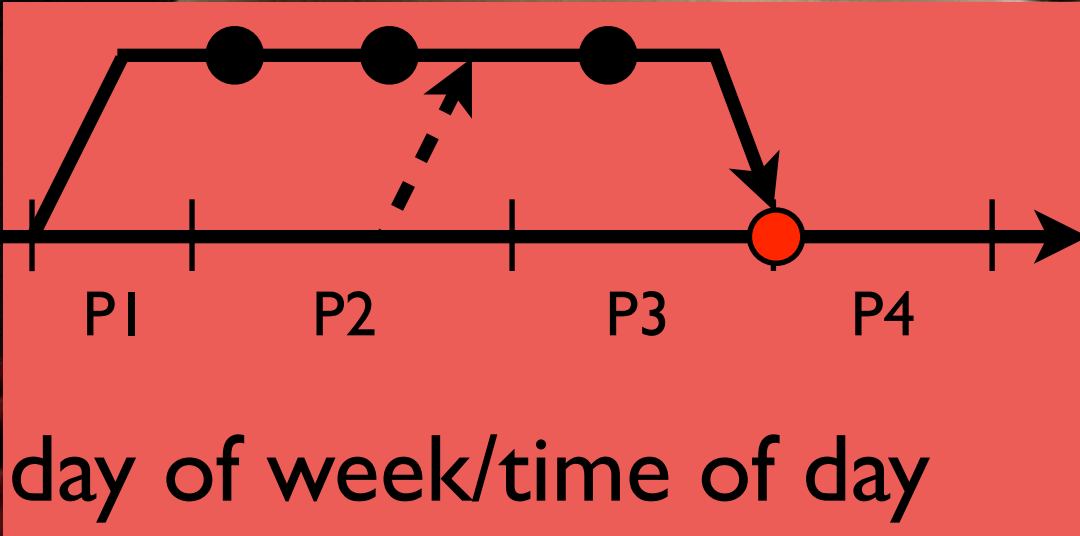
VS. VS. VS. VS. VS. VS.



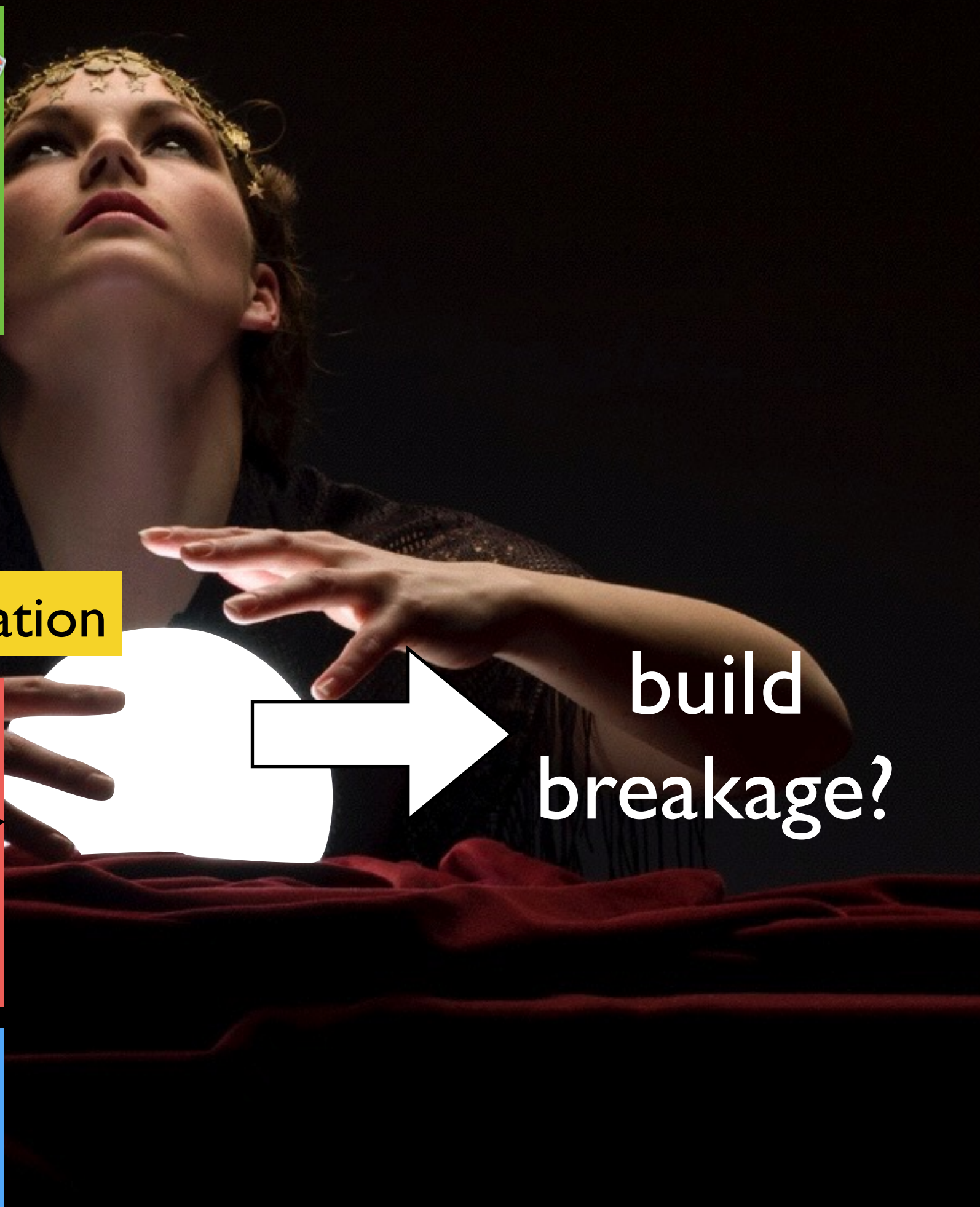
local vs. remote

code/file churn
commit vs. merge


feature vs. bug fix vs. integration



platform/operating system
...



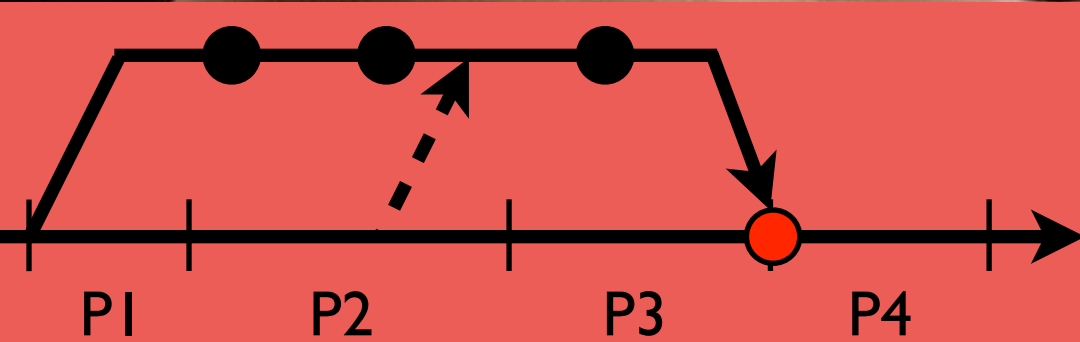
VS. VS. VS. VS. VS. VS.



local vs. remote

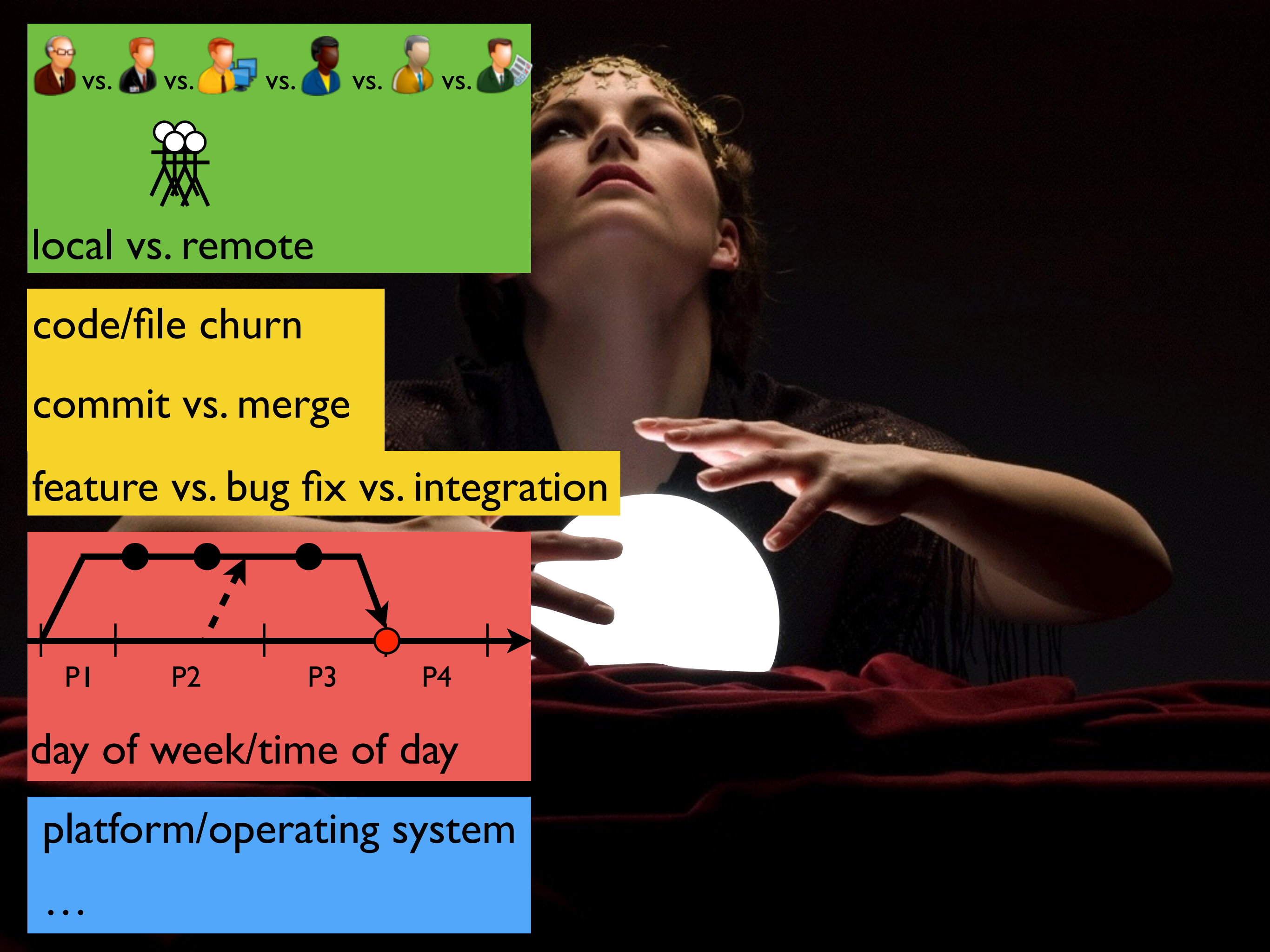
code/file churn
commit vs. merge

feature vs. bug fix vs. integration




day of week/time of day

platform/operating system
...

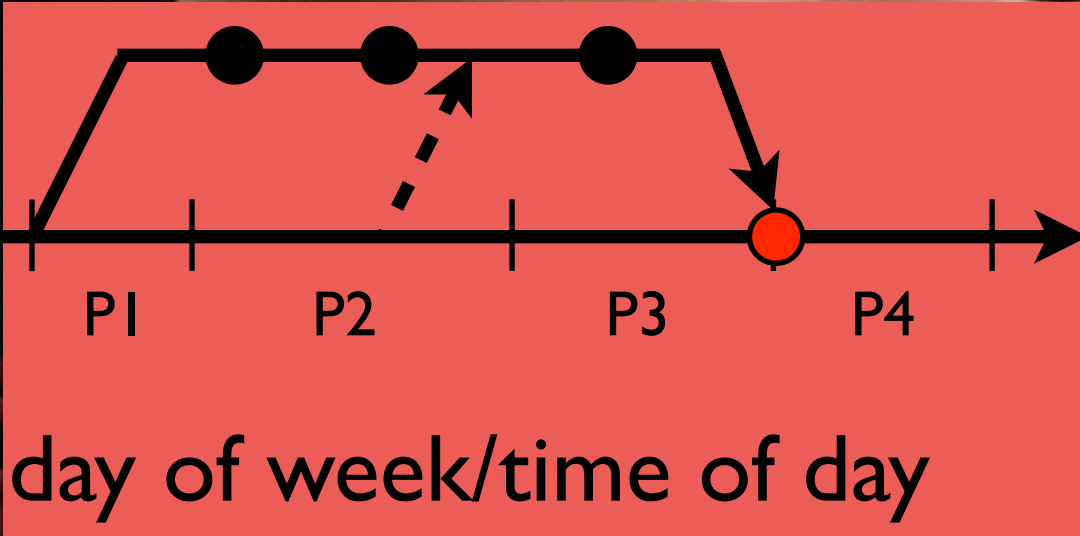


VS. VS. VS. VS. VS. VS.



local vs. remote

code/file churn
commit vs. merge
feature vs. bug fix vs. integration



platform/operating system
...

build resources
needed (VMs,
cloud nodes, etc.)

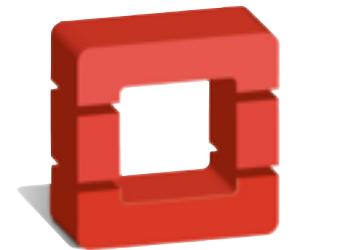
expected
build time

build
refactorings

...



How to Schedule CI Builds?



openstack™

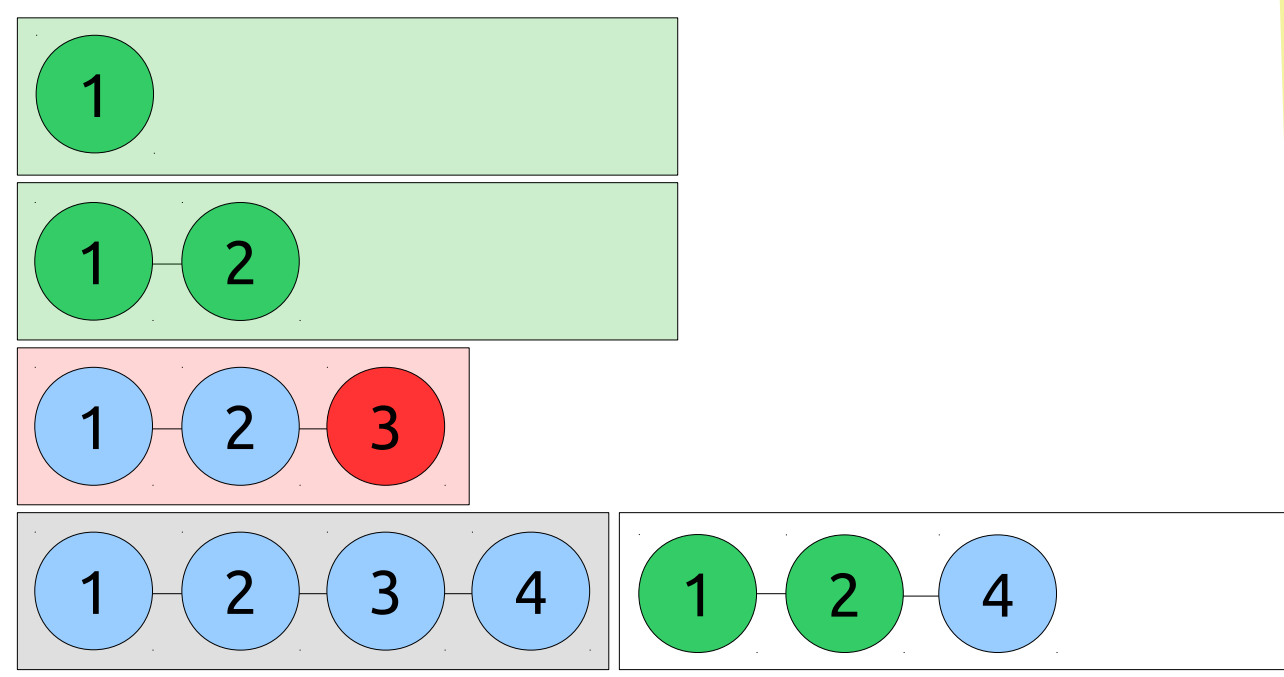
How to Schedule CI Builds?

serial execution is slow,
but easy to blame build



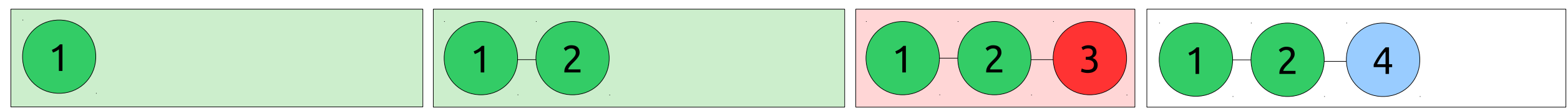


How to Schedule CI Builds?



parallel execution is fast, but awkward to blame build

serial execution is slow, but easy to blame build





integrating code changes



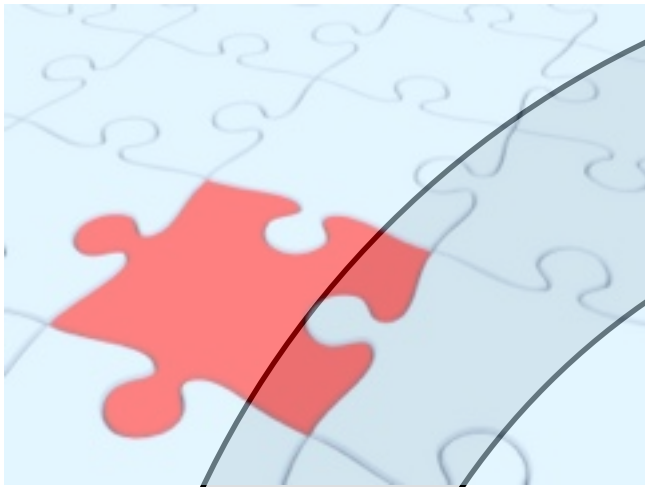
building/testing (CI)



releasing to the user



deploying a new release



integrating code changes

building/testing (CI)



releasing to the user

deploying a new release



What are deployment best practices?



What are deployment best practices?



What are deployment best practices?



What are deployment best practices?



What are deployment best practices?



What are deployment best practices?



What are deployment best practices?



When is canary a
success?



What are deployment best practices?

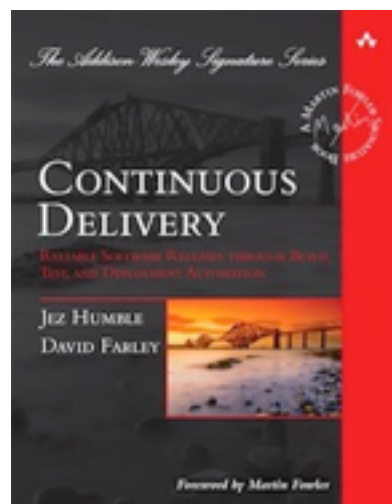
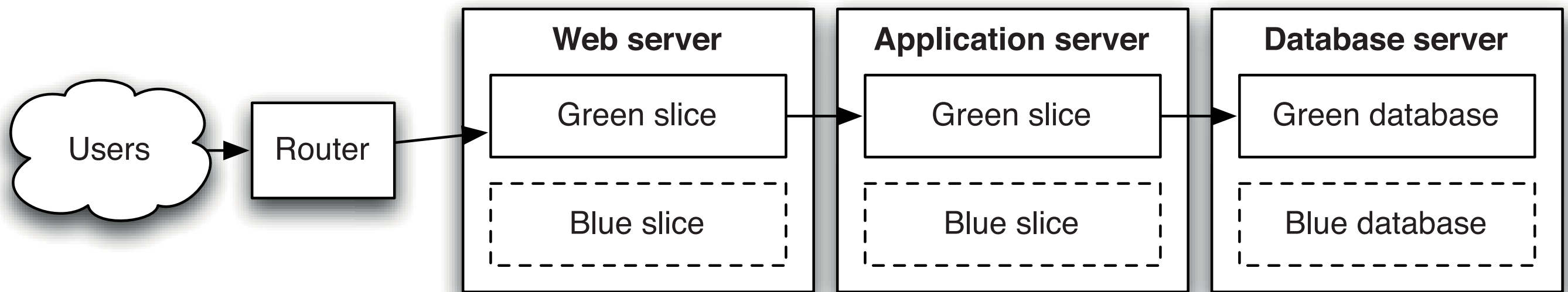
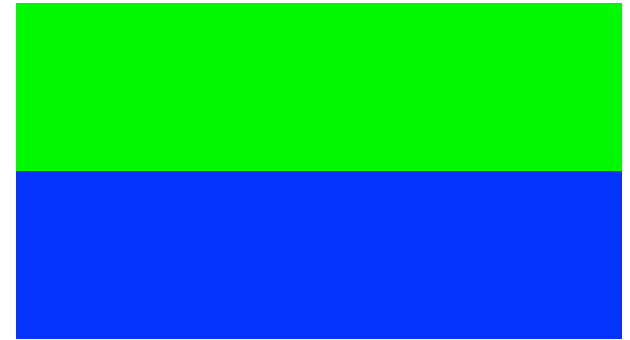


When is canary a success?

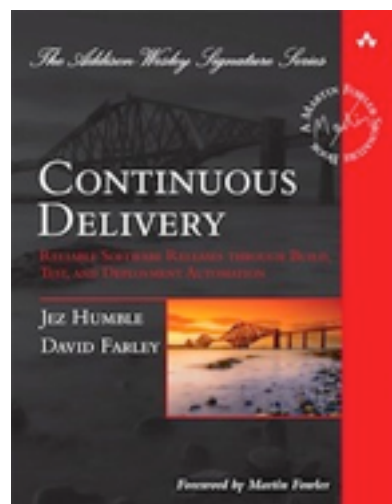
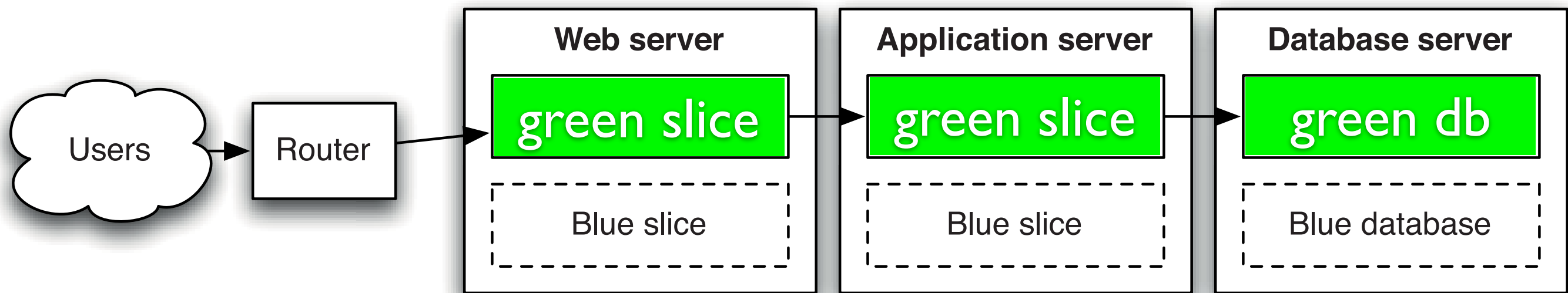
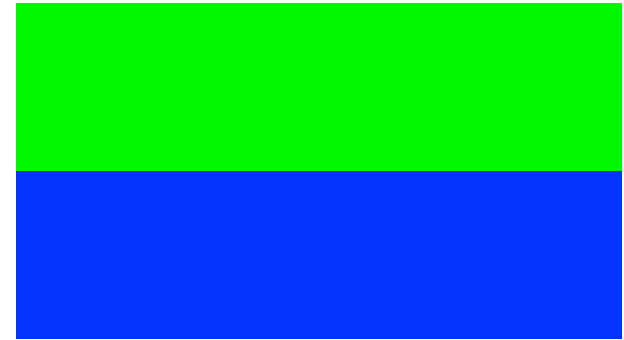
Which users should receive the canary?



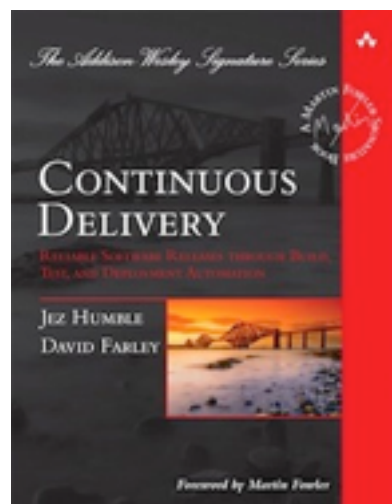
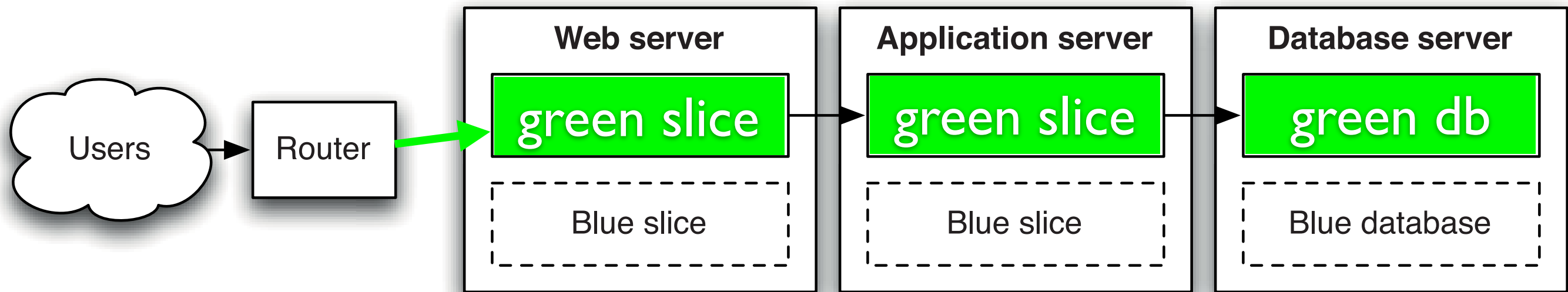
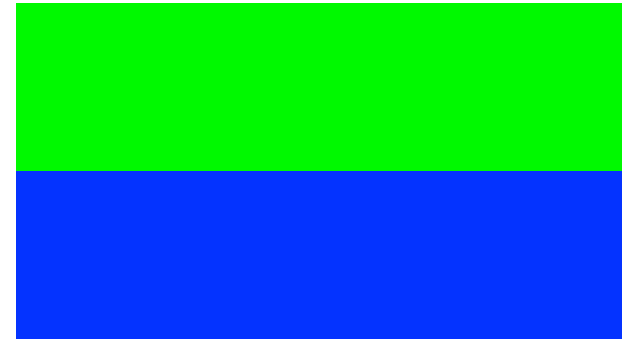
Blue-Green Deployment



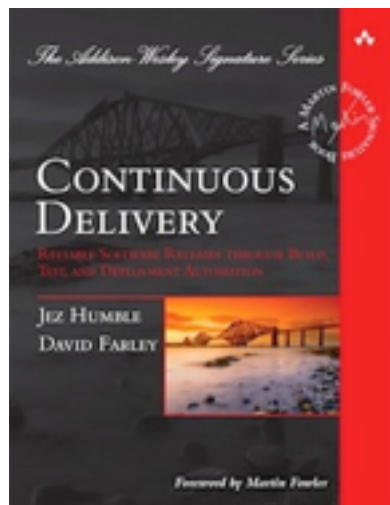
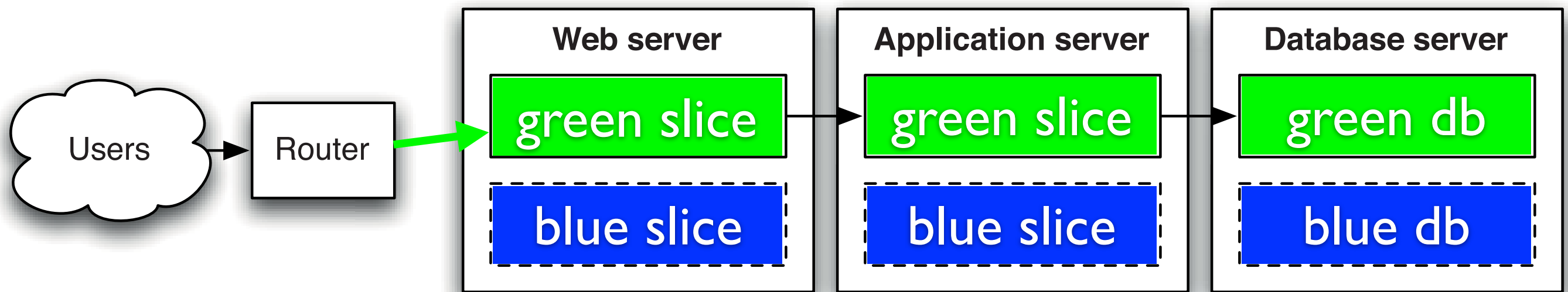
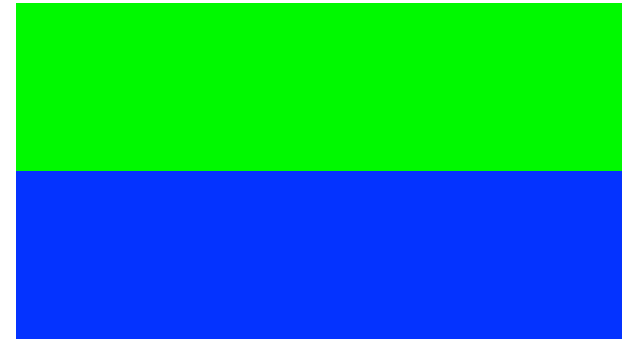
Blue-Green Deployment



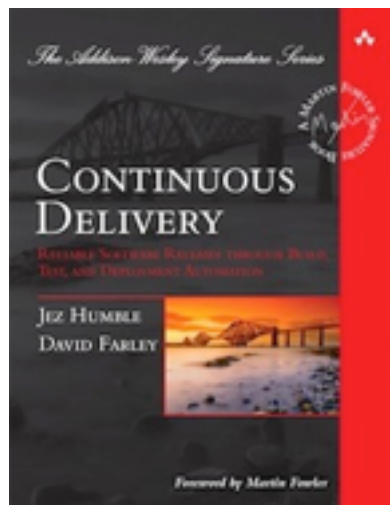
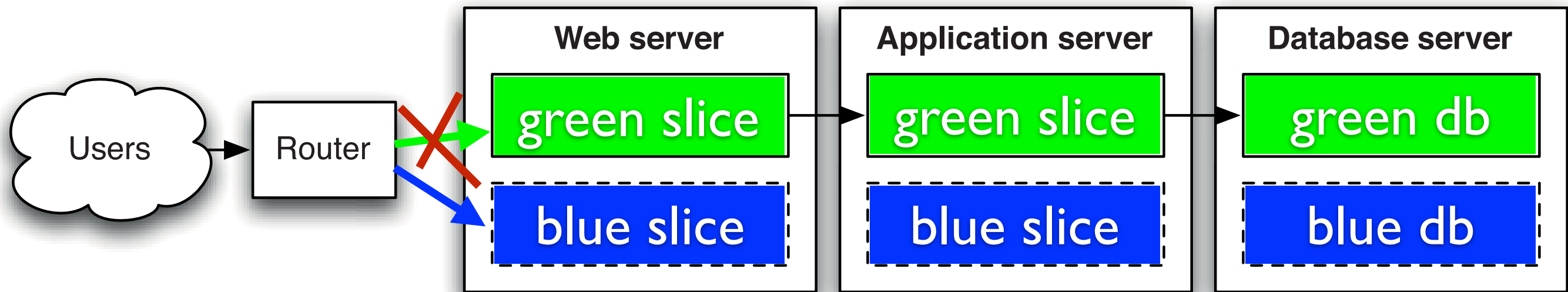
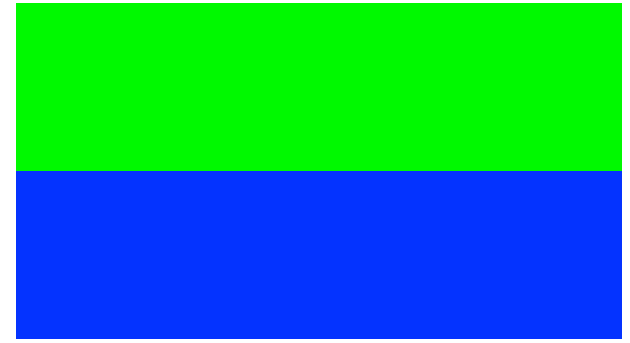
Blue-Green Deployment



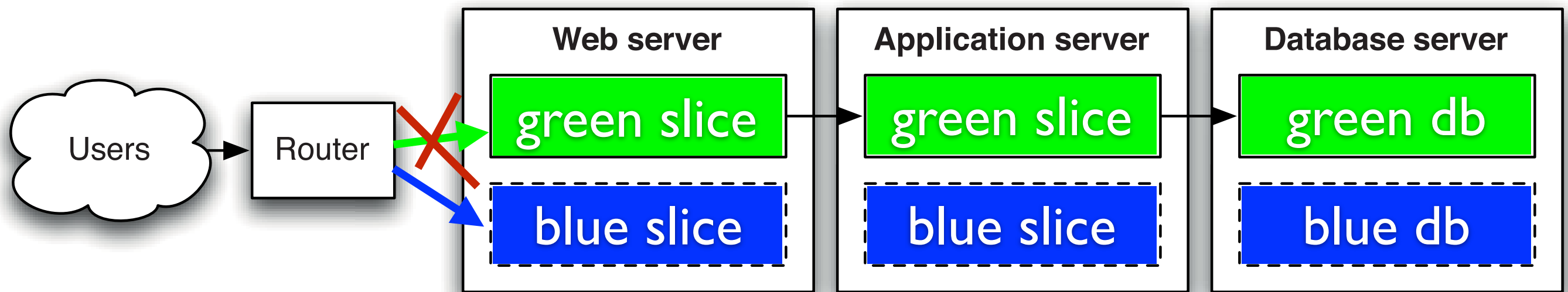
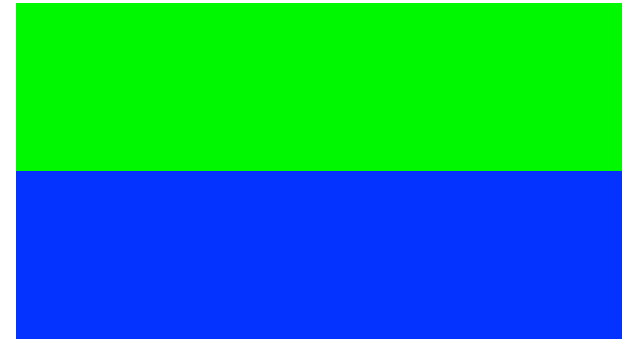
Blue-Green Deployment



Blue-Green Deployment

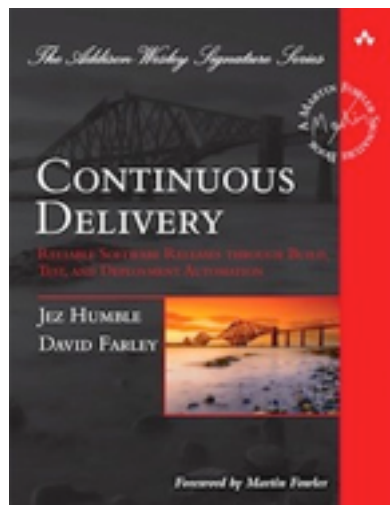


Blue-Green Deployment



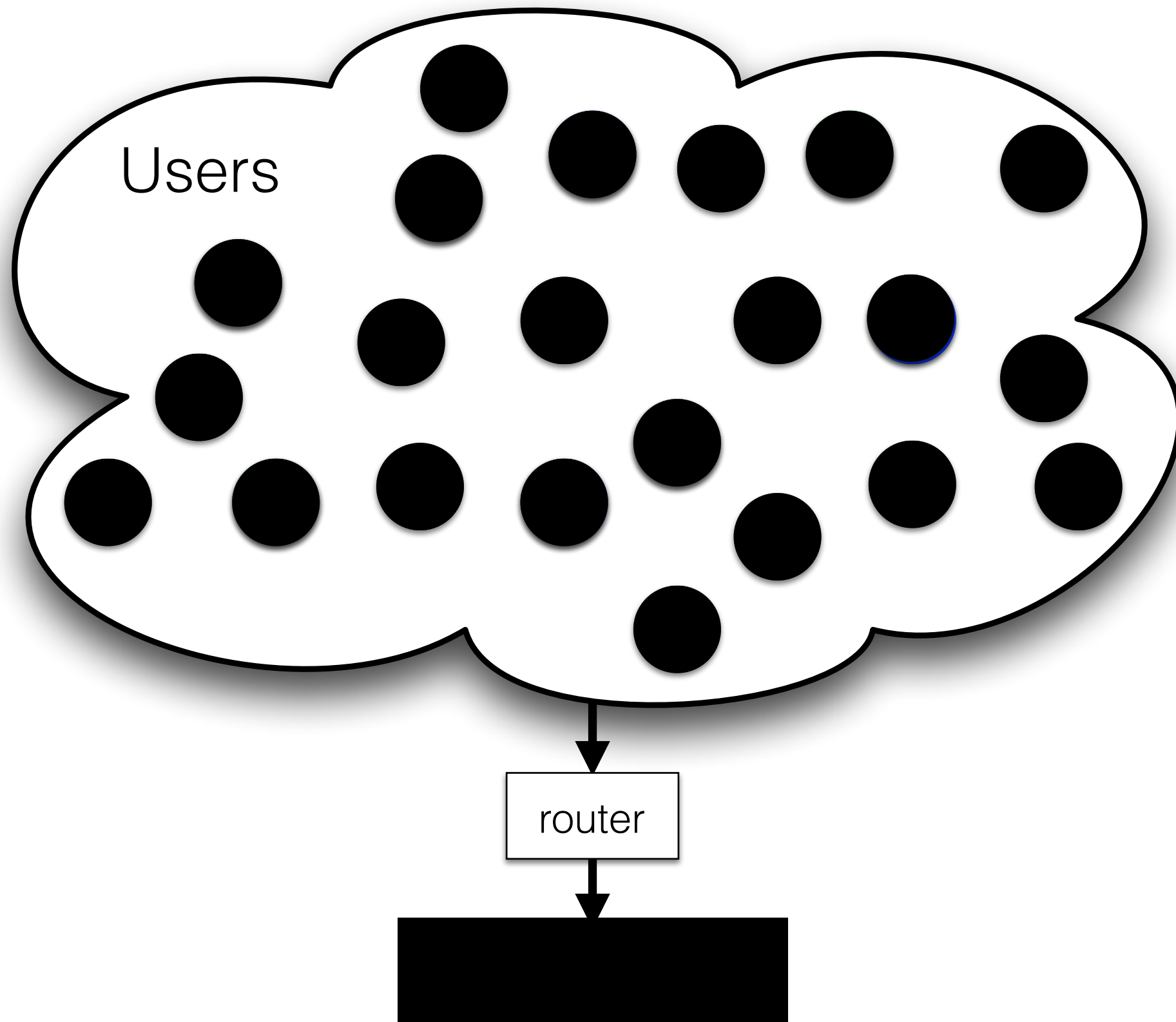
RYOR: give the right people the power (and tools) to kick off deployments

Linked 



A/B Deployment

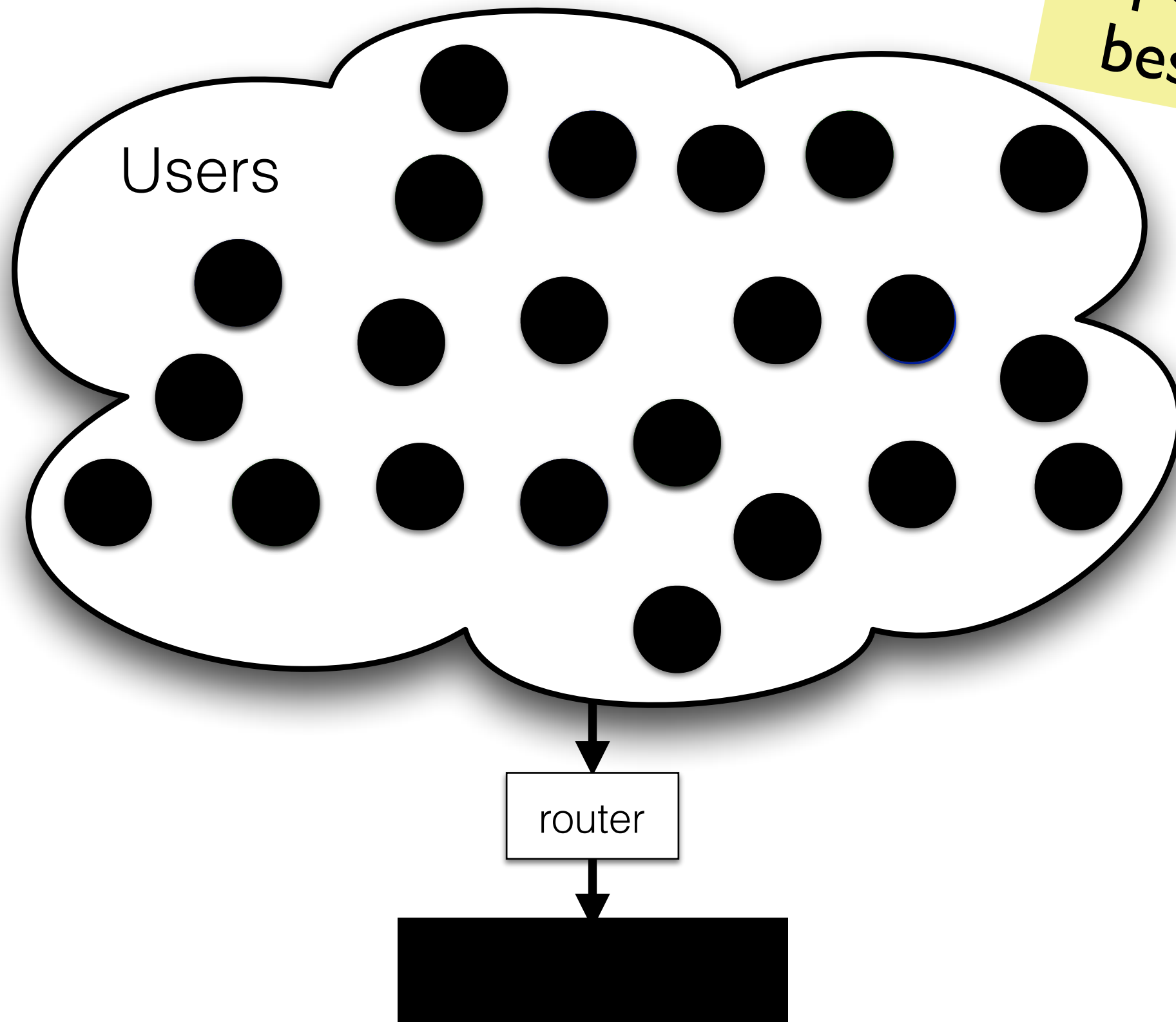
A/B



A/B Deployment

A/B

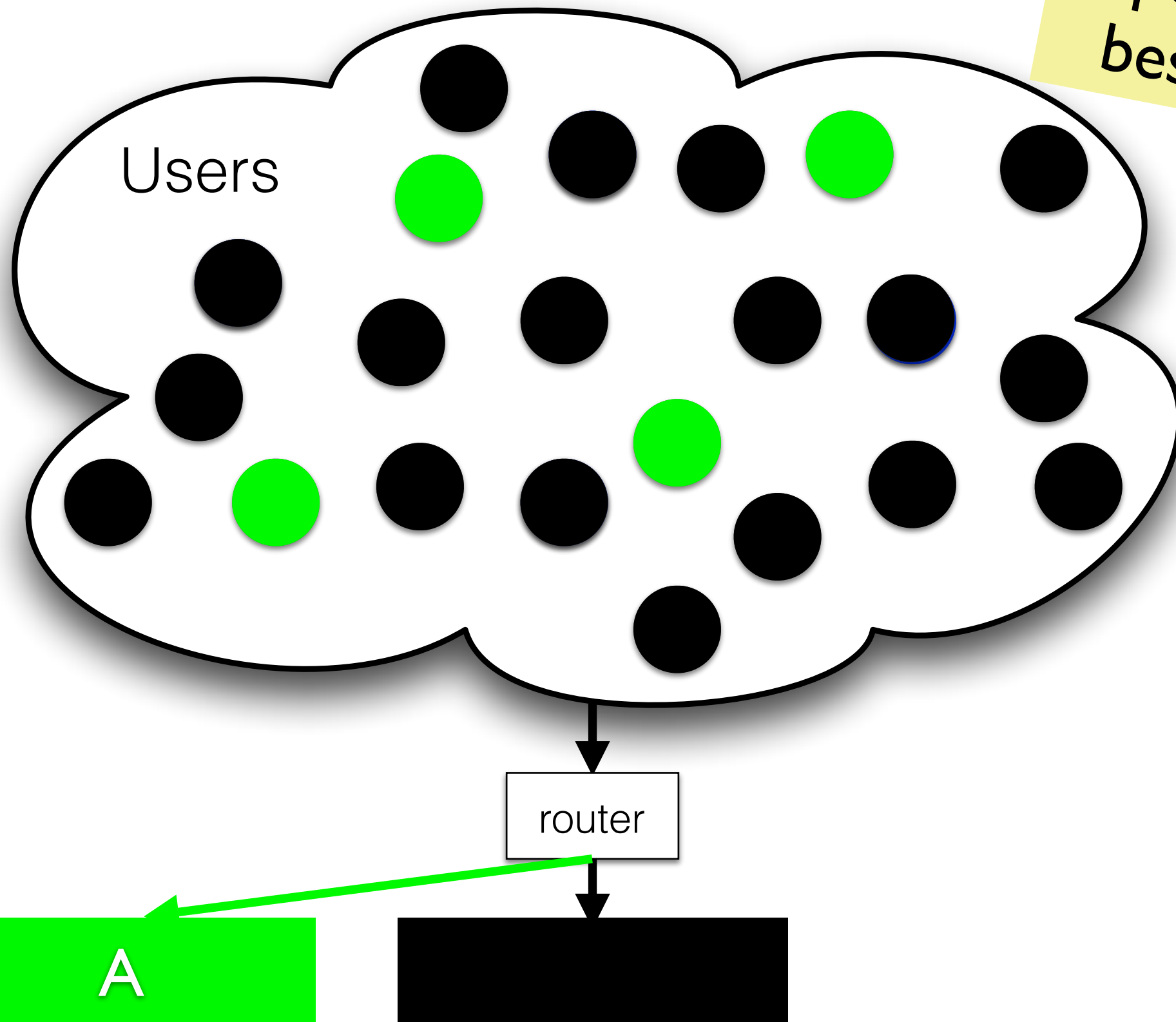
short-term
experiment to find
best alternative



A/B Deployment

A/B

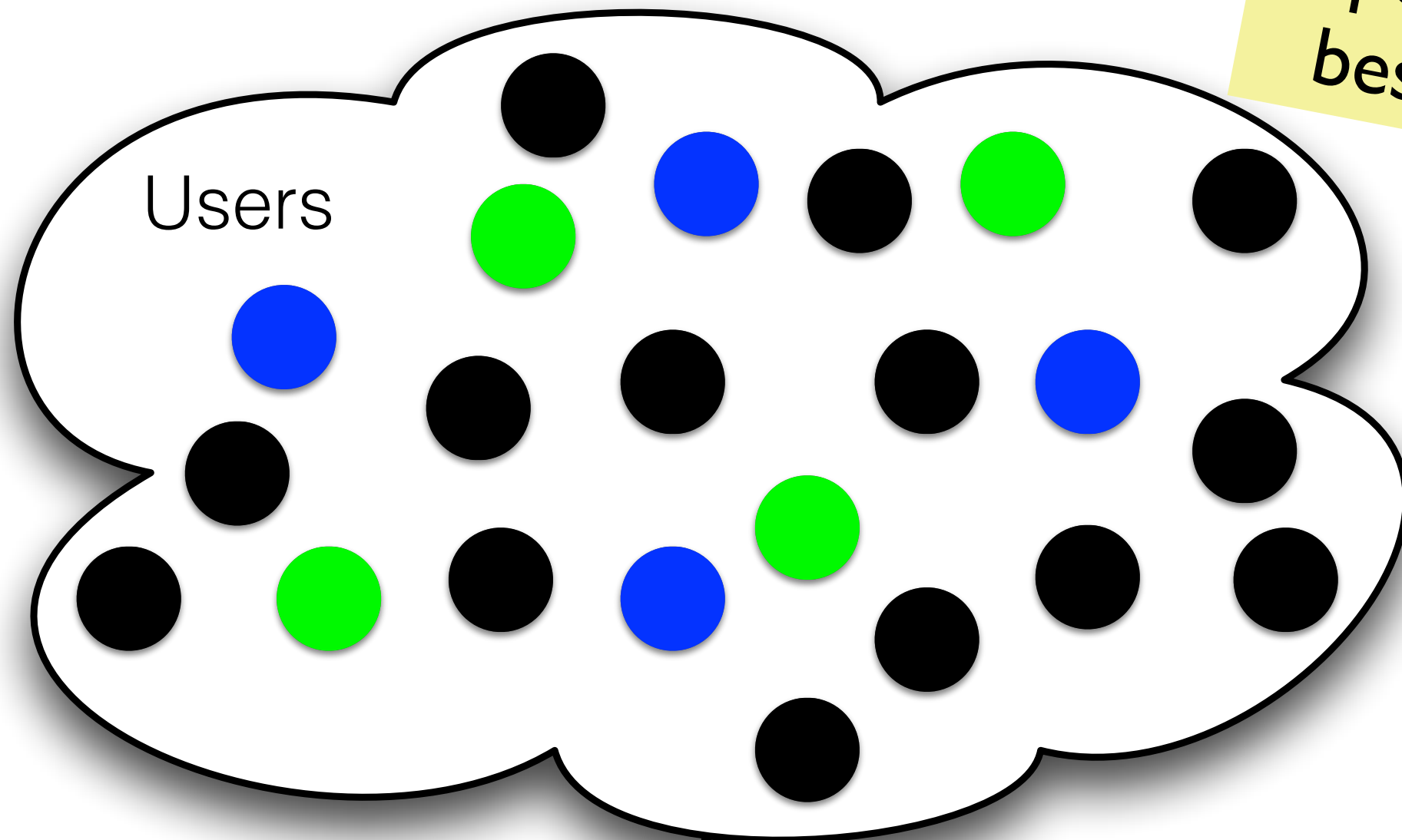
short-term
experiment to find
best alternative



A/B Deployment

A/B

short-term
experiment to find
best alternative



router

A

B






```
# Install PostgreSQL server and client
include_recipe "postgresql::server"
include_recipe "postgresql::client"

# Make postgresql_database resource available
include_recipe "database::postgresql"

# Create database for Rails app
db = node["practicingruby"]["database"]
postgresql_database db["name"] do
  connection(
    :host      => db["host"],
    :port      => node["postgresql"]["config"]["port"],
    :username  => db["username"],
    :password  => db["password"],
  )
end
```



Infrastructure code smells?

```
# Install PostgreSQL server and client
include_recipe "postgresql::server"
include_recipe "postgresql::client"

# Make postgresql_database resource available
include_recipe "database::postgresql"

# Create database for Rails app
db = node["practicingruby"]["database"]
postgresql_database db["name"] do
  connection(
    :host      => db["host"],
    :port      => node["postgresql"]["config"]["port"],
    :username  => db["username"],
    :password  => db["password"],
  )
end
```





Co-evolution of Infrastructure and Source Code - An Empirical Study

Yujuan Jiang, Bram Adams
MCIS lab Polytechnique Montreal, Canada
Email: {yujuan.jiang, bram.adams}@polymtl.ca

MSR 2015



Co-evolution of Infrastructure and Source Code - An Empirical Study

Yujuan Jiang, Bram Adams
MCIS lab Polytechnique Montreal, Canada
Email: {yujuan.jiang, bram.adams}@polymtl.ca

MSR 2015

Does Your Configuration Code Smell?

Tushar Sharma, Marios Fragkoulis and Diomidis Spinellis
Dept of Management Science and Technology
Athens University of Economics and Business
Athens, Greece
{tushar,mfg,dds}@aueb.gr

MSR 2016





integrating code changes



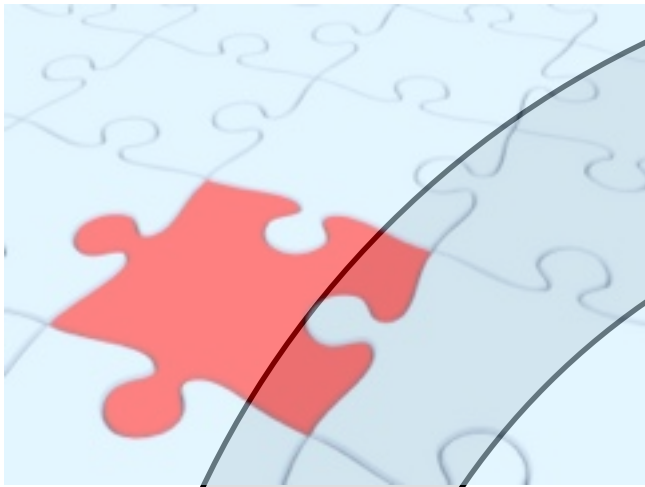
building/testing (CI)



releasing to the user



deploying a new release



integrating code changes



building/testing (CI)



releasing to the user




deploying a new release




Is the
release in good
shape?



A man with a surprised expression is shown from the chest up, sitting at a desk. He has a beard and is wearing a dark, patterned sweater. His eyes are wide open, and his mouth is slightly agape. On the desk in front of him are a white computer mouse and a keyboard. To his right, there is a cluttered desk area with various papers, a box of Crayons, and other items. In the top right corner, there is a small, open cardboard box. Two speech bubbles are overlaid on the image, containing text.

Is the
release in good
shape?


Was the release a
success?

A man with a surprised expression is shown from the chest up, sitting at a desk. He has light brown hair and a beard, and is wearing a dark green and black patterned sweater. His eyes are wide open, and his mouth is slightly agape. He is looking directly at the camera. In the background, there is a wooden desk with a white computer mouse and a keyboard. To the right, there is a white storage unit with various items inside, including a box of snacks. A small, open cardboard box is floating in the upper right corner of the image. Three speech bubbles are overlaid on the image, containing text related to a software release.

Is the
release in good
shape?

Was the release a
success?

The release is
botched, how can we
roll back?

A man with a surprised expression is sitting at a desk with a computer. He is wearing a green and black patterned sweater. There are three speech bubbles overlaid on the image, each containing a question. In the top right corner, there is a small icon of an open cardboard box.

Is the
release in good
shape?

Was the release a
success?

The release is
botched, how can we
roll back?

What's the optimal cycle
time for us?





Is the
release in good
shape?

Was the release a
success?

How will a
faster cycle impact
our software
quality?

The release is
botched, how can we
roll back?

What's the optimal cycle
time for us?



Is the
release in good
shape?

Was the release a
success?

The release is
botched, how can we
roll back?

How will a
faster cycle impact
our software
quality?

What's the optimal cycle
time for us?

What best practices
should we adopt?

Rolling Back with Run-time Feature Toggles



Rolling Back with Run-time Feature Toggles



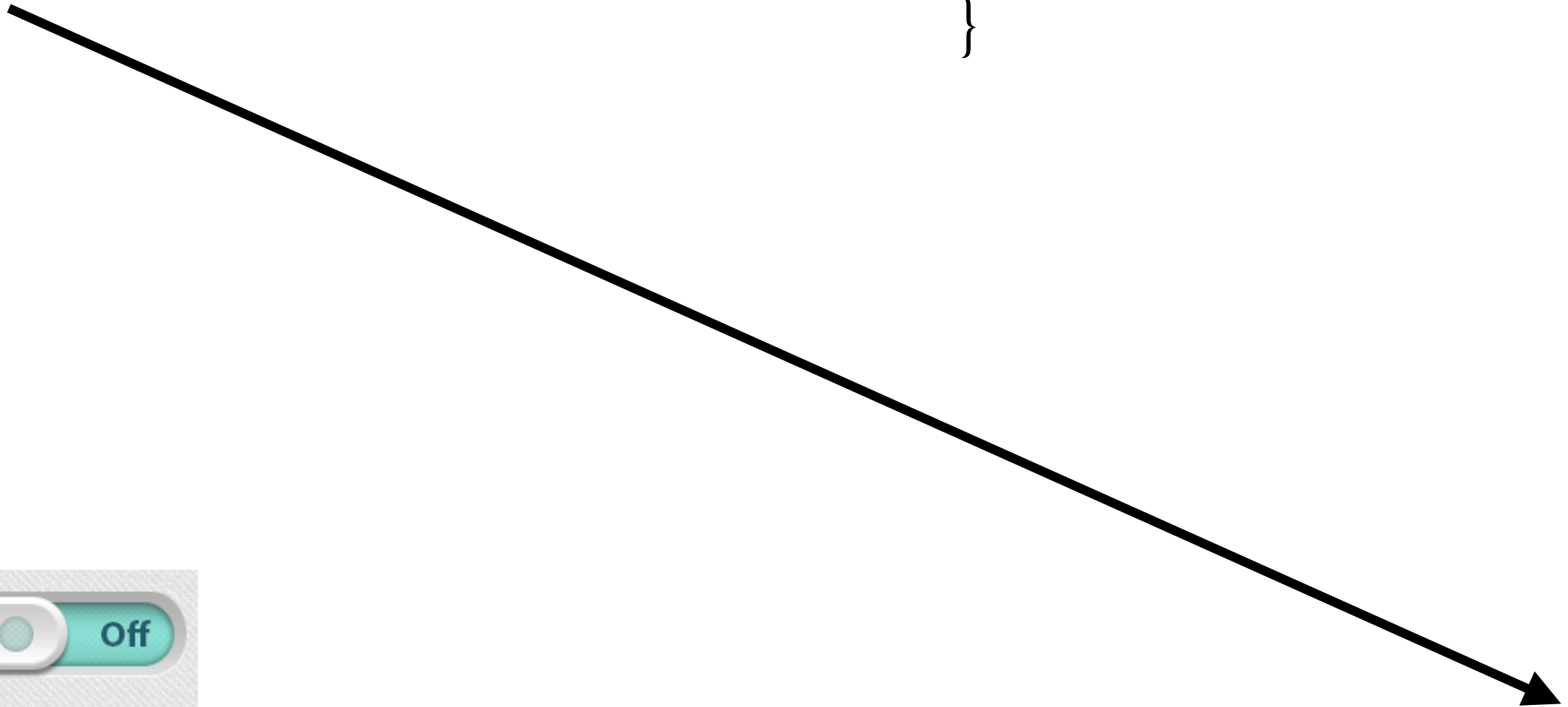
```
if(new_feature_on==true){  
    /* code of new feature */  
}
```



Rolling Back with Run-time Feature Toggles



```
if(new_feature_on==true){  
    /* code of new feature */  
}
```



time

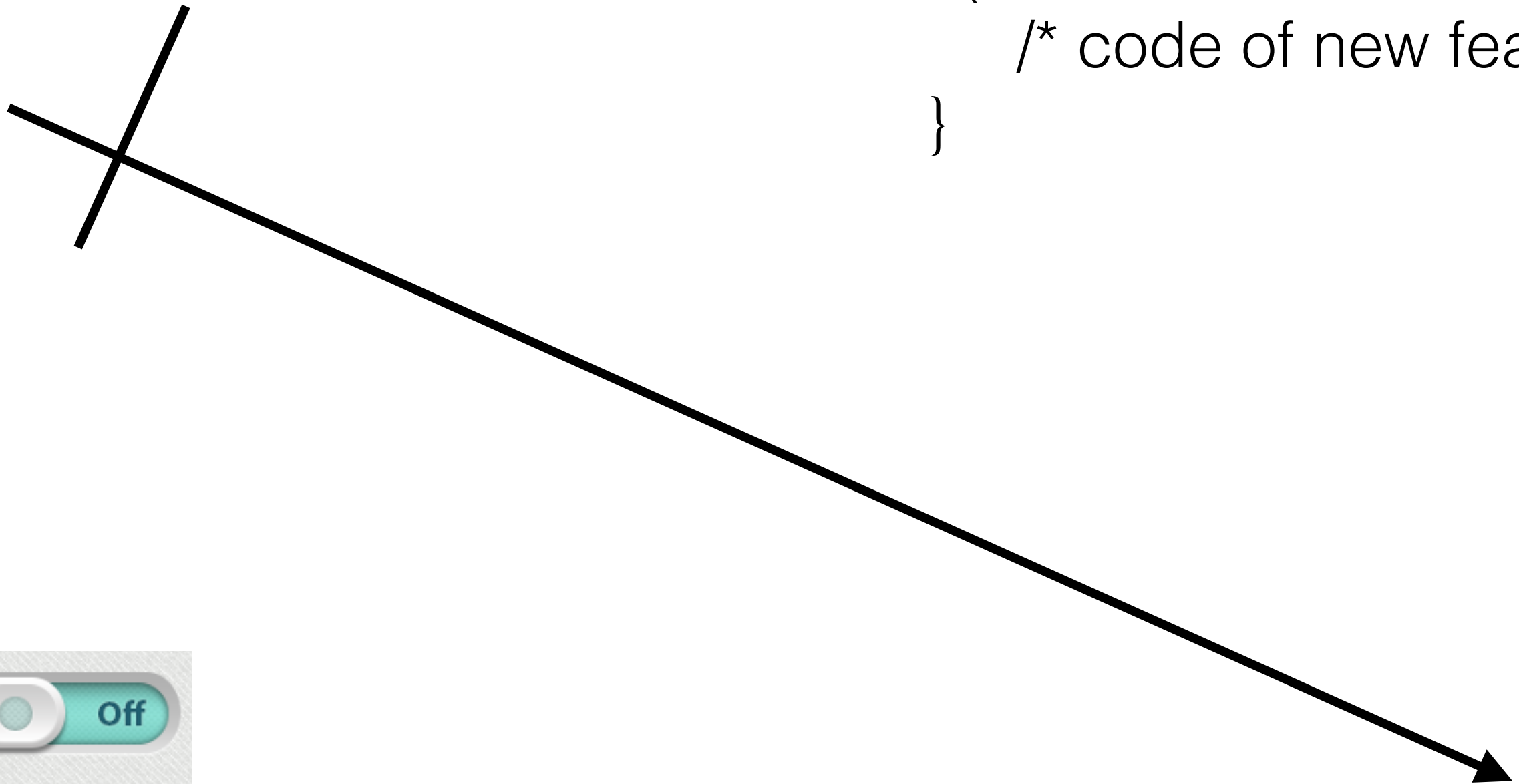


Rolling Back with Run-time Feature Toggles



testing

```
if(new_feature_on==true){  
    /* code of new feature */  
}
```



time



Rolling Back with Run-time Feature Toggles



testing

```
if(new_feature_on==true){  
    /* code of new feature */  
}
```

ON

time



Rolling Back with Run-time Feature Toggles



testing

```
if(new_feature_on==true){  
    /* code of new feature */  
}
```

ON

release

time



Rolling Back with Run-time Feature Toggles



testing

```
if(new_feature_on==true){  
    /* code of new feature */  
}
```

ON

release

ROLLBACK

time



Rolling Back with Run-time Feature Toggles



testing

```
if(new_feature_on==true){  
    /* code of new feature */  
}
```

ON

release

OFF

ROLLBACK

time



Oct 17, 2014 10:38:02 PM -kwierso@gmail.com 92c87e95915e

92c87e95915e	WK	Backed out changeset ba0bb4f26680 (1080055) for
c8a9a49c5c10	WK	Backed out 3 changesets (1075644) for Android bu
043792c7257a	RV	1084158 - Update pdf.js to version 1.0.907. r=bdah
3b8d0c8b28a5	MB	1084384: support alternate phone number values fr
e8a01f5feb55	MB	1079941: implement LoopContacts.search to allow
ba0bb4f26680	DT	1080055: When recreating a browser restore the is
c44f001ea6c1	JC	1075644 - Wait for page to fully load in context mei
ca0dff936c2	JC	1075644 - Move more things to after Gecko start; r
47264e561dce	JC	1075644 - Reduce Gecko thread priority at very sta
ae71da76c733	MC	1075531 - Part 3: Match page titles by regex in test
9eb089d7c2b9	MC	1075531 - Part 2: Check for url in place of page titl
8882bf98a437	MC	1075531 - Part 1: Return String from ToolbarComp
75c3e79b9c7c	MB	1013989: update the localization note to match the
808971467f9f	RG	1079811 - A new call won't start if the outgoing cal
410e7c3c6a1e	CB	Merge m-c to fx-team
7be62fa11c20	MH	1063586 - Audit tab related XML files to use new te
67b723fc33f1	RB	1080701 - TabMirror needs to be updated to work
bb58a5fc381b	MT	1024807 - Add "Report Site Issue" menu item (for l
f0e0f05aa2d3	AK	1082970 - Sync no longer shows login failed notific
16627f2d7729	HS	1081203 - Remove dead #social-toolbar-item CSS
...and 2 more		

Linux opt	B Cpp Jit1 Jit2 Mn N Wr X M(1 2 3 4 5 bc1 bc2 bc3 dt oth) M-e10s(1 2 3 4 5 bc1 bc2 bc3) R(C Cipc J R Ripc Ru) T(c d g1 s tp) W(1 2 3 4)
Linux pgo	B Cpp Cpp Jit1 Jit1 Jit2 Jit2 Mn Mn Wr Wr X X M(1 1 2 2 3 3 4 4 5 5 bc1 bc1 bc2 bc2 bc3 bc3 dt dt oth oth) M-e10s(1 1 2 2 3 3 4 4 5 5 bc1 bc1 bc2 bc2 bc3 bc3) R(C C Cipc Cipc J J R R Ripc Ripc Ru Ru) T(c c d d g1 g1 s s tp tp) W(1 1 2 2 3 3 4 4)
Linux debug	B Cpp Jit1 Jit2 Mn X M(1 2 3 4 5 bc1* bc2 bc3 dt1 dt2 dt3 oth) M-e10s(1 2 3 4 5) R(C J R)
Linux x64 opt	B Bn Cpp H Jit1 Jit2 Mn N V Wr X M(1 2 3 4 5 bc1 bc2 bc3 dt oth) M-e10s(1 2 3 4 5 bc1 bc2 bc3) R(C J R) T(c d g1 o o s tp) W(1 2 3 4)
Linux x64 pgo	B Cpp Cpp Jit1 Jit1 Jit2 Jit2 Mn Mn Wr Wr X X M(1 1 2 2 3 3 4 4 5 5 bc1 bc1* bc2 bc2 bc3 bc3 dt dt oth oth) M-e10s(1 1 2 2 3 3 4 4 5 5 bc1 bc1 bc2 bc2 bc3 bc3) R(C C J J R R) T(c c d d g1 g1 o o o o s s tp tp) W(1 1 2 2 3 3 4 4)
Linux x64 asan	Bd Bo Cpp Cpp Jit1 Jit1 Jit2 Jit2 Nd No X X* M(1 1 2 2 3 3 4 4 5 5 bc1 bc1 bc2 bc2 bc3 bc3 dt dt oth oth) M-e10s(1 1 2 2 3 3 4 4 5 5 bc1 bc1* bc2 bc2 bc3 bc3) R(C C J J R R)
Linux x64 debug	B Bn Cpp Jit1 Jit2 Mn S X M(1 2 3 4 5 bc1 bc2 bc3 dt1 dt2 dt3 oth) M-e10s(1 2 3 4 5) R(C J R)
OS X 10.6 opt	Cpp Cpp Jit Jit X X M(1 1 2 2 3 3 4 4 5 5 bc1 bc1 bc2 bc2 bc3 bc3 dt dt oth oth) M-e10s(bc1 bc1 bc2 bc2 bc3 bc3) R(C C J J R R) T(c c d d g1 g1 o o s s tp tp)
OS X 10.6 debug	Cpp Jit Mn X M(1 2 3 4 5 bc1 bc2 bc3 dt1 dt2 dt3 oth) R(C J R)
OS X 10.8 opt	B Bn N X X M(1 1 2 2 3 3 4 4 5 5 bc1 bc1 bc2 bc2 bc3 bc3 dt dt oth oth) M-e10s(bc1 bc1 bc2 bc2 bc3 bc3) R(C C J J R R) T(c c d d g1 g1 o o s s tp tp)
OS X 10.8 debug	B Bn Mn X M(1 2 3 4 5 bc1 bc2 bc3 dt1 dt2 dt3 oth) R(C J R)
Windows XP opt	B Bn Cpp Jit Mn N X M(1 2 3 4 5 bc1 bc2 bc3 dt oth) R(C J

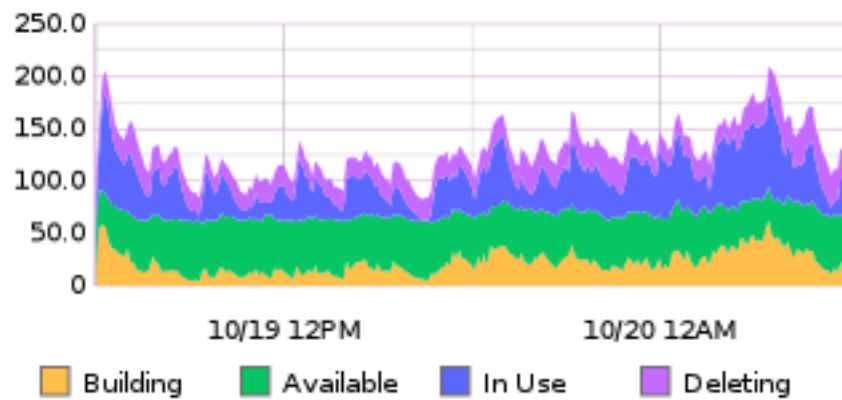
Monitoring the Release Progress

What to Monitor?

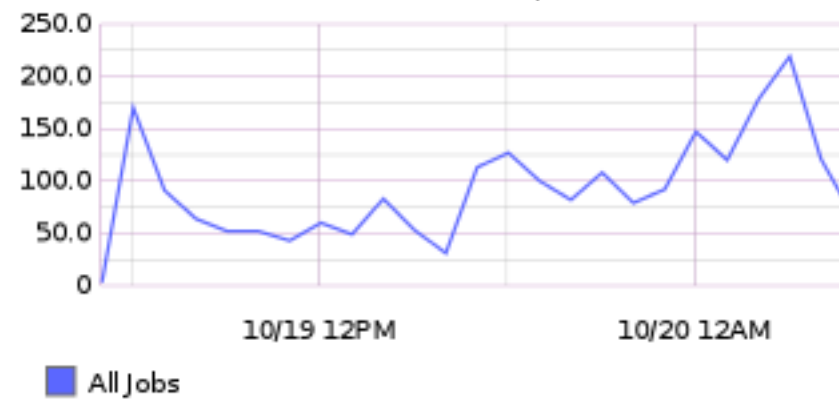


Job Stats

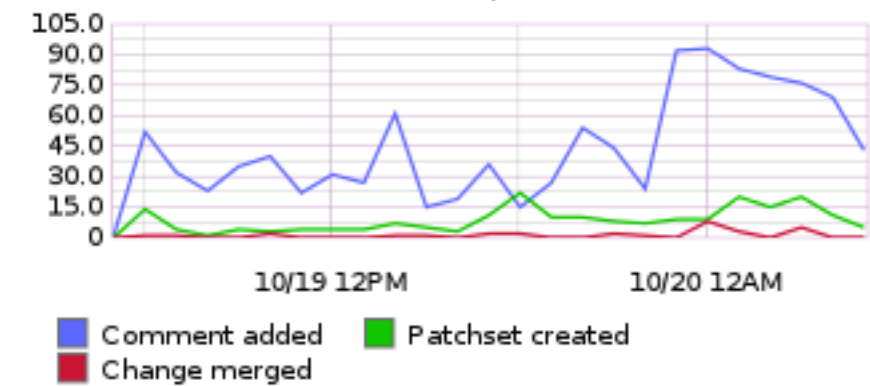
Test Nodes



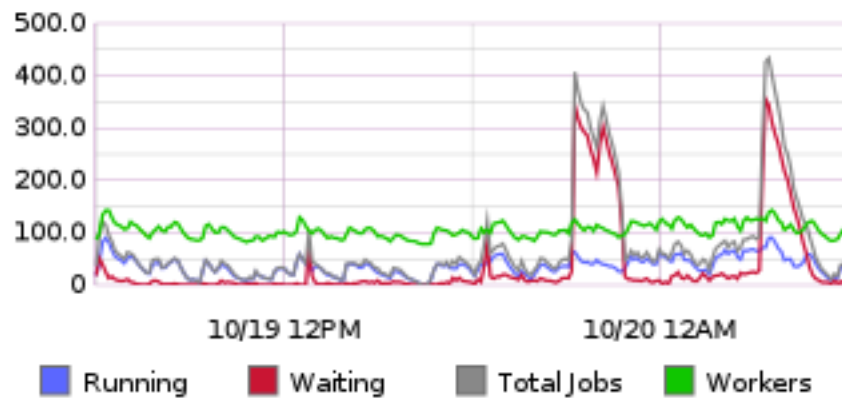
Zuul Jobs Launched (per Hour)



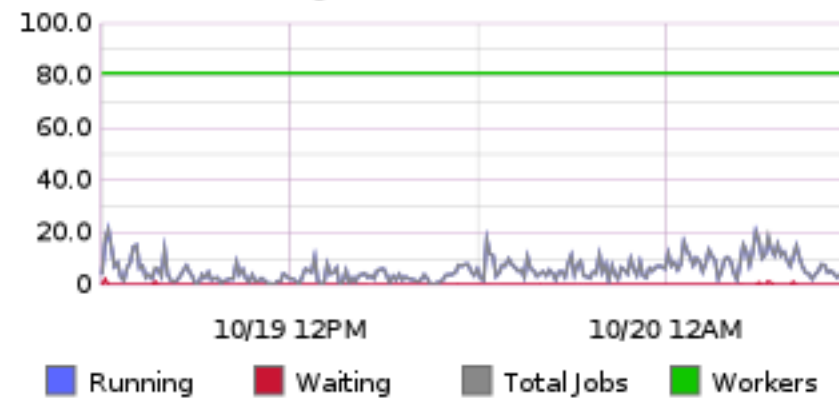
Gerrit Events (per Hour)



Zuul Job Queue



Logstash Job Queue




Zuul info

Zuul version: 2.0.0.284

Last reconfigured: Sat Oct 18 2014 18:55:58 GMT-0400 (EDT)





Metrics are important,
but people optimize for what you
measure or show them, so choose
your KPI carefully!
[John O'Duinn]







#changed code lines

#open bugs



coding time

#releases



#changed code lines

#open bugs



coding time

#releases

%passing tests

time from commit to release



#changes/release

time to interaction

#cherry-picks/release

bug rates

time between release

crash rates

app size

What can **you**
do for release
engineering?

What can **you**
do for release
engineering?

What can
release
engineering
do **for you**?

What do you mean, Dr. Adams?!
These **poor students' research**
is impacted by the release
engineering process? They are
just doing empirical research on
large software data!





1. All Releases are Equal



1. All Releases are Equal



Look at Project X!
It had a crazy number of
post-release bugs in this six-
month time window!



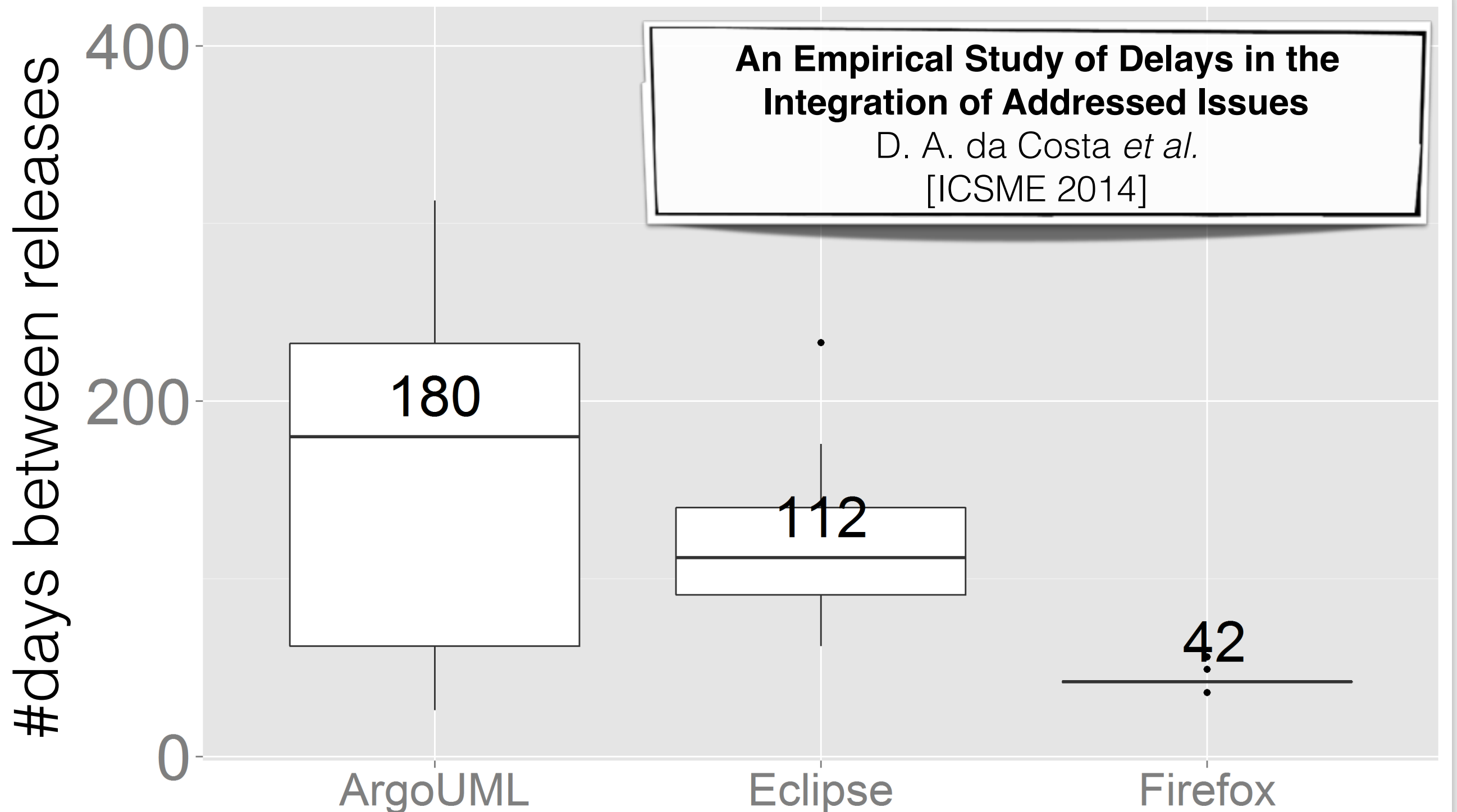


Look at Project X!
It had a crazy number of
post-release bugs in this six-
month time window!

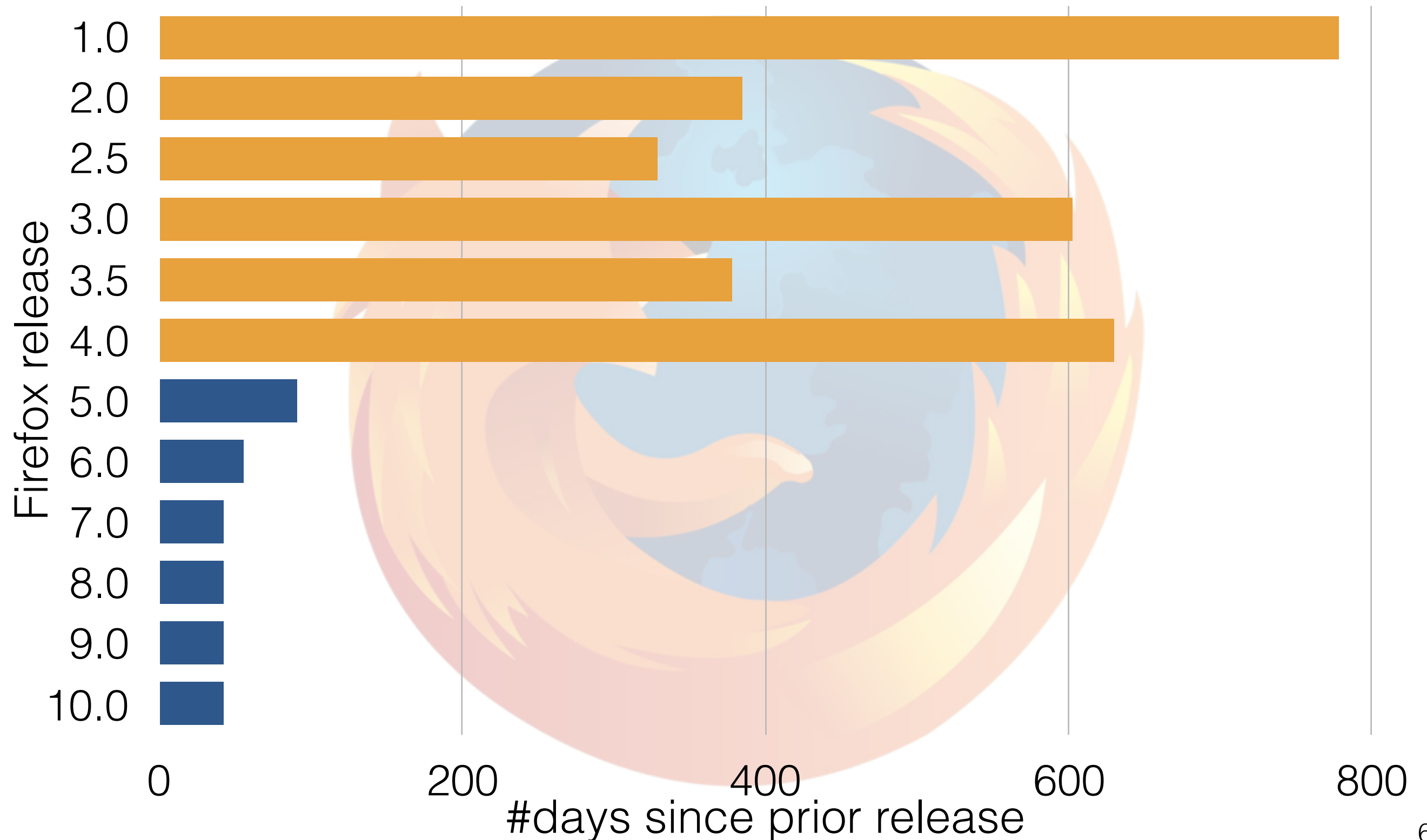


Well, Project X releases
every 6 weeks, so
you're counting several
releases...

Release Cycles Vary among Popular Studied Systems



..., Even Within Systems!



Features can be Older than the Cycle Time

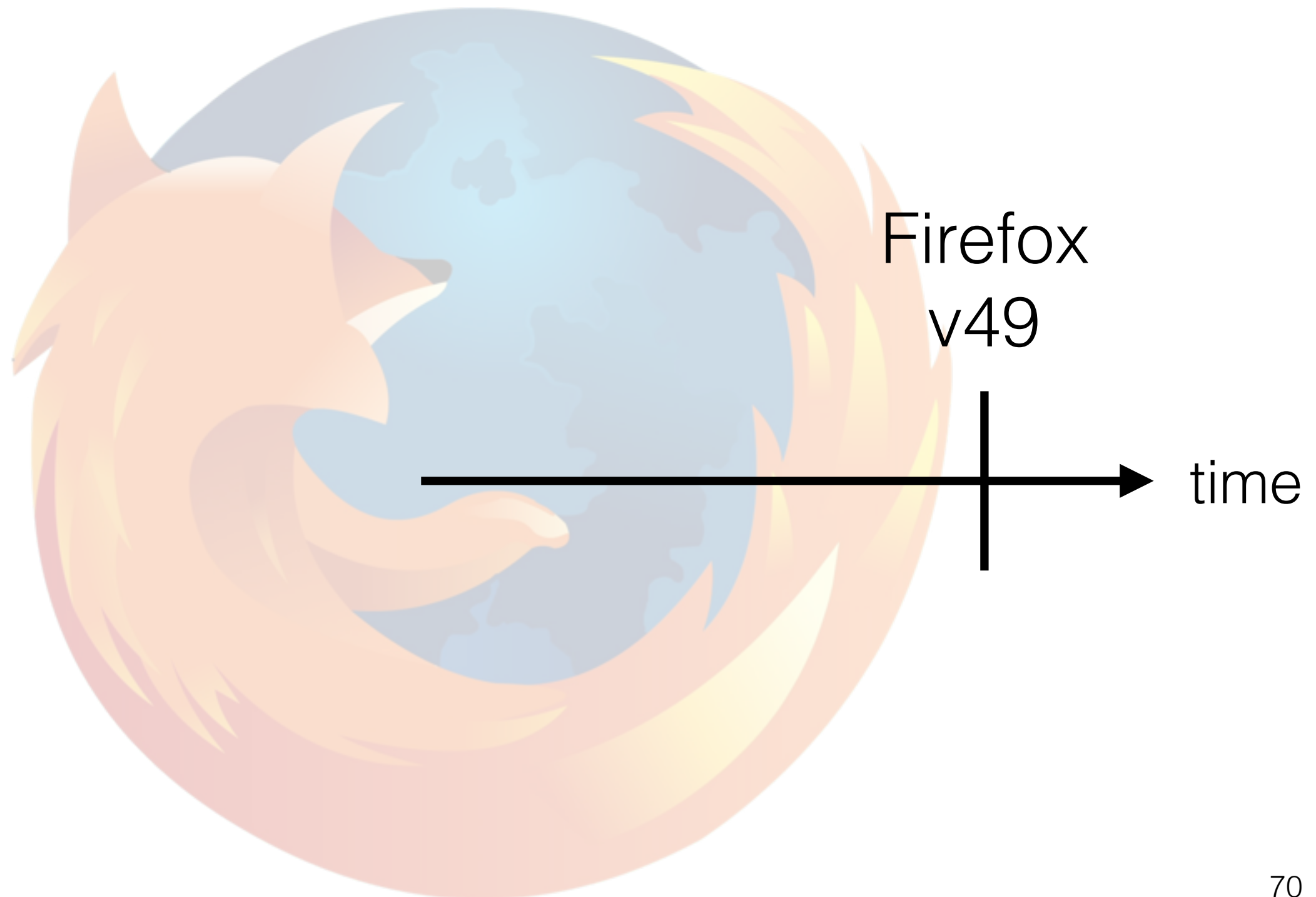


Features can be Older than the Cycle Time

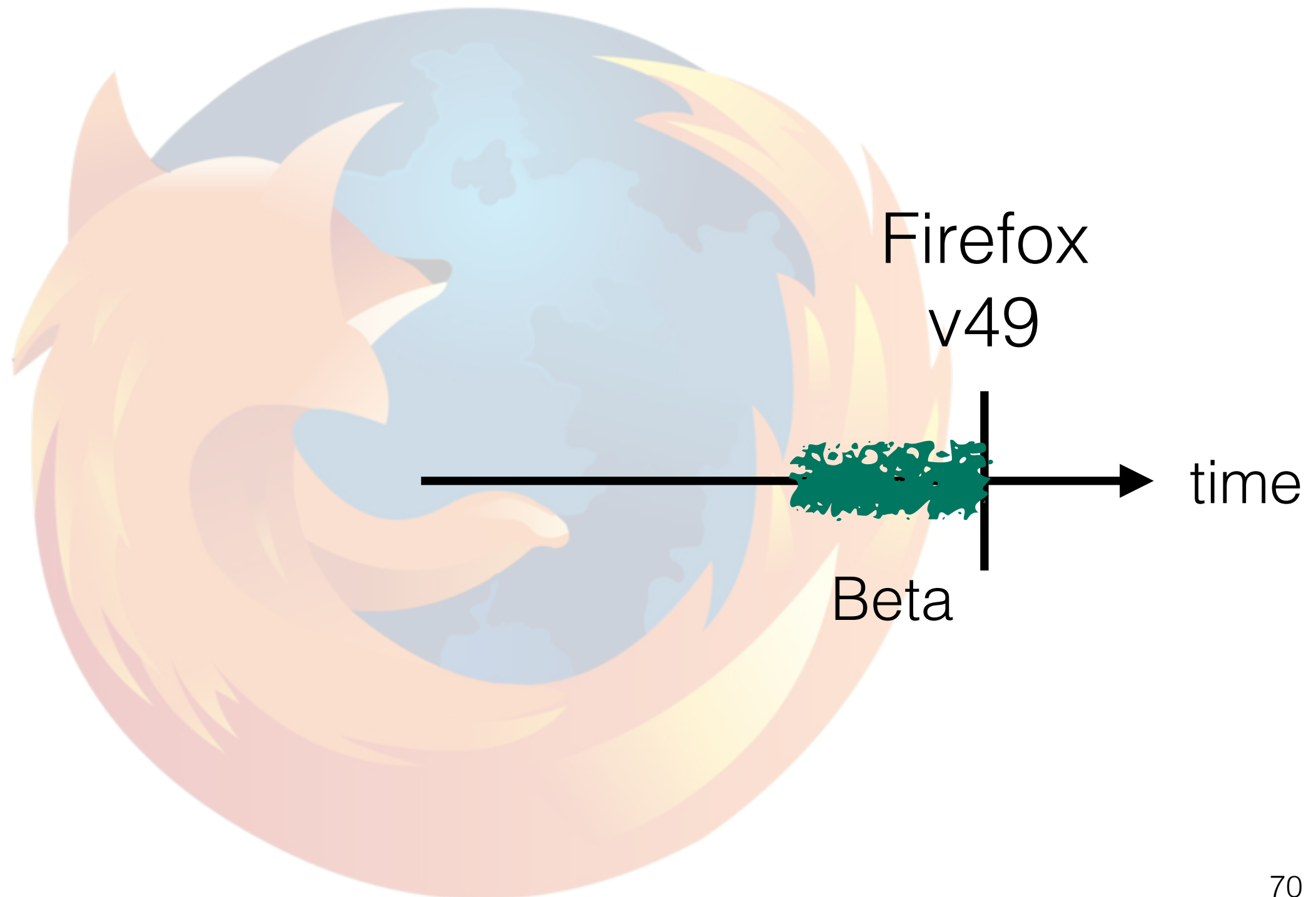


Firefox
v49

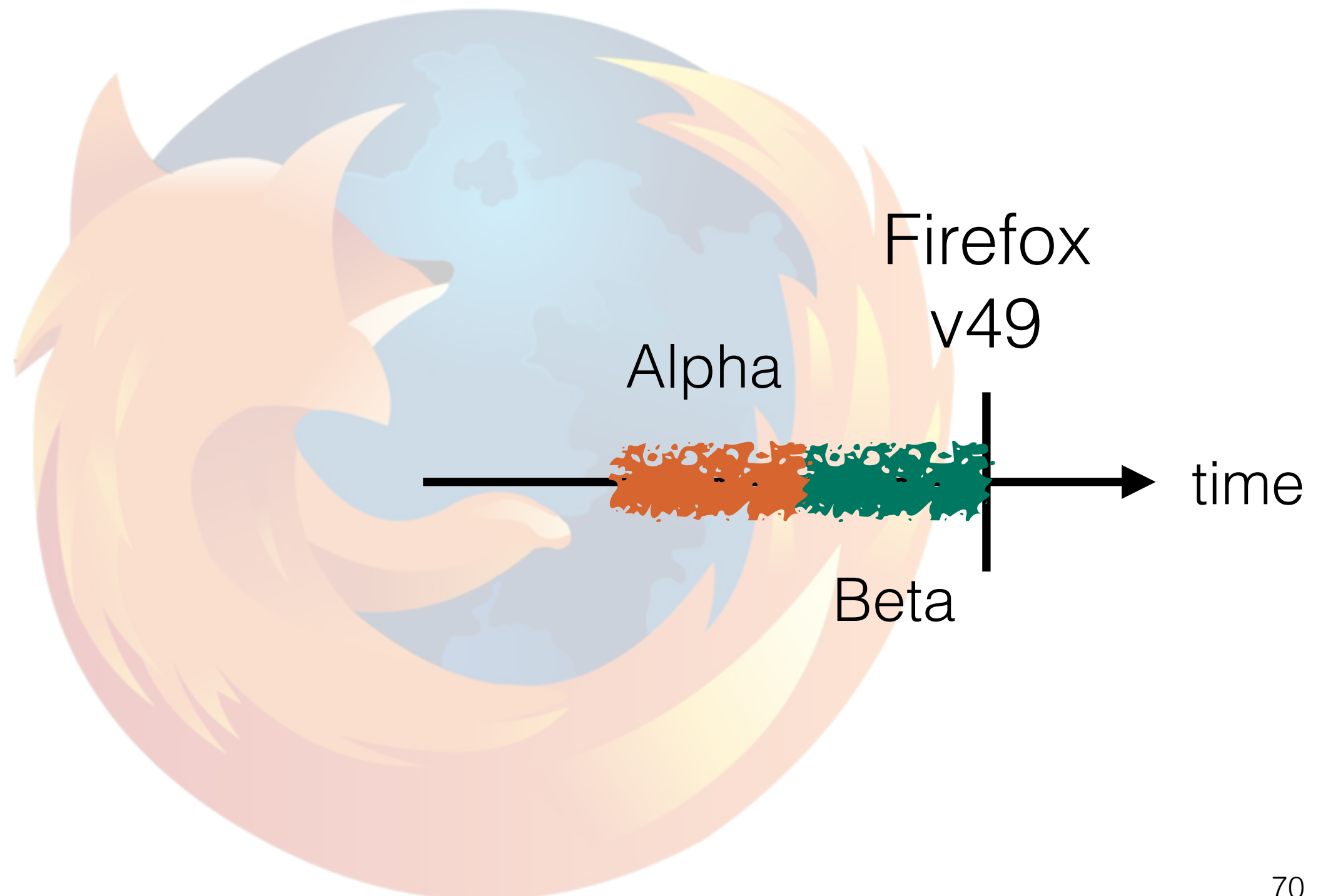
Features can be Older than the Cycle Time



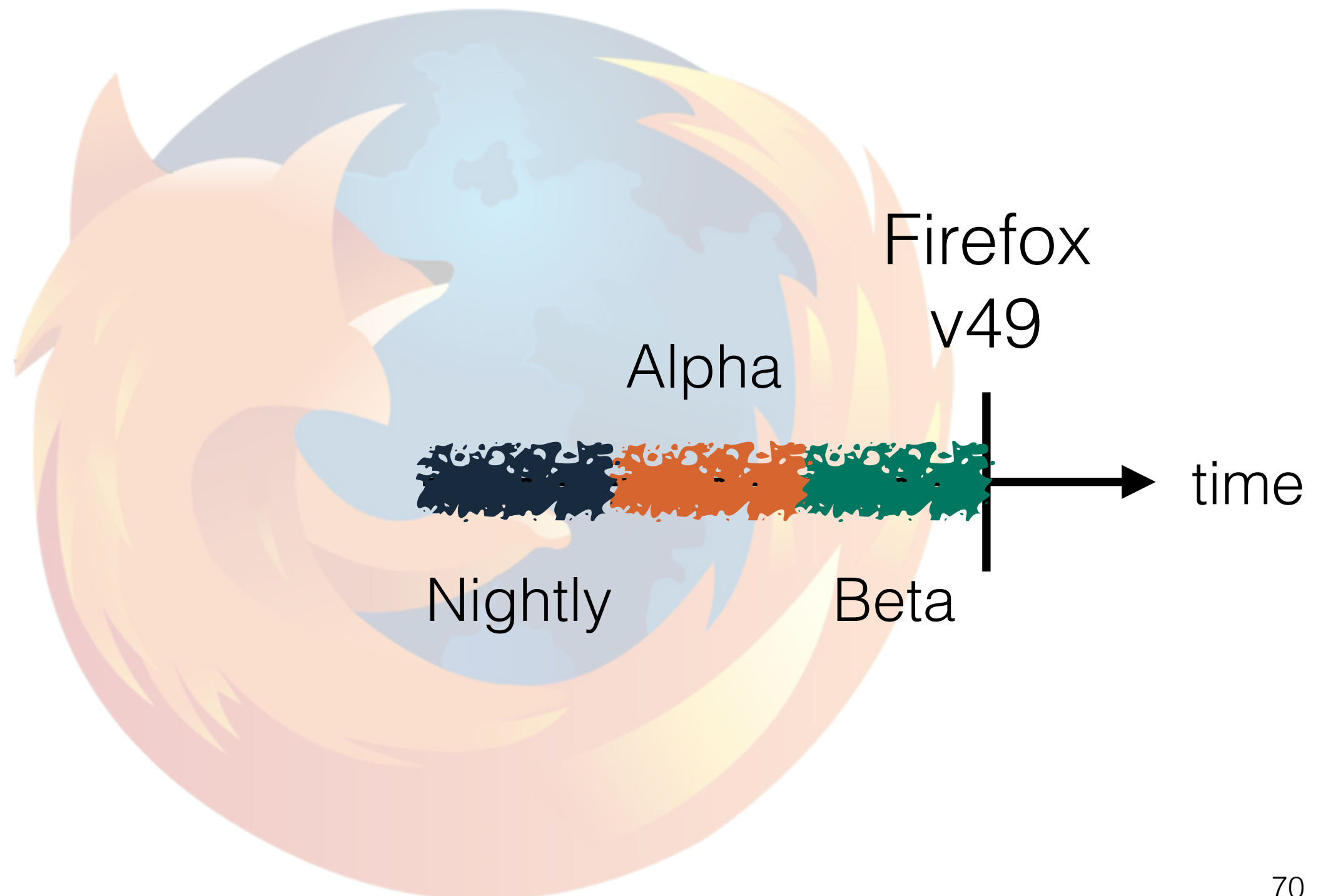
Features can be Older than the Cycle Time



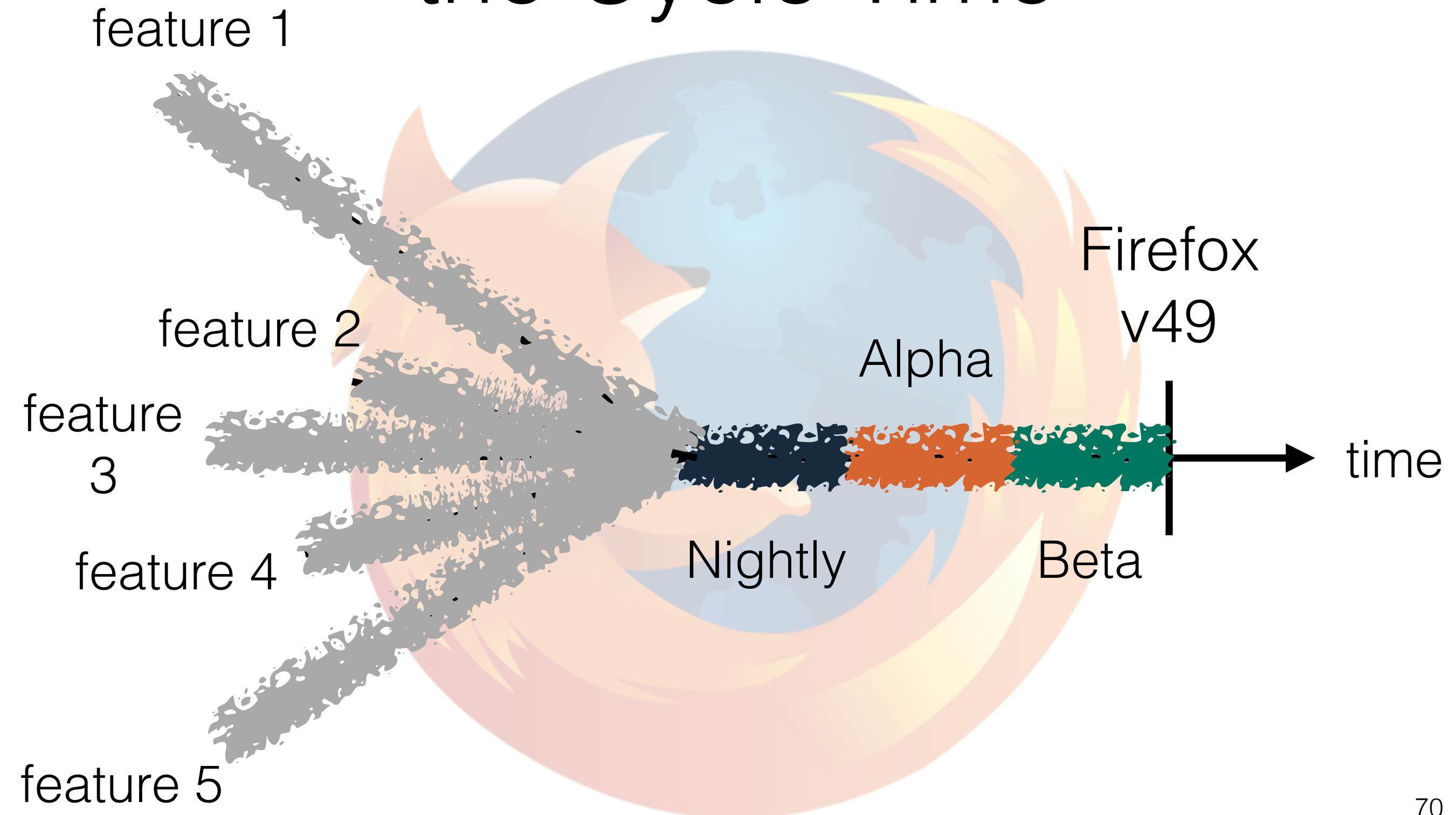
Features can be Older than the Cycle Time

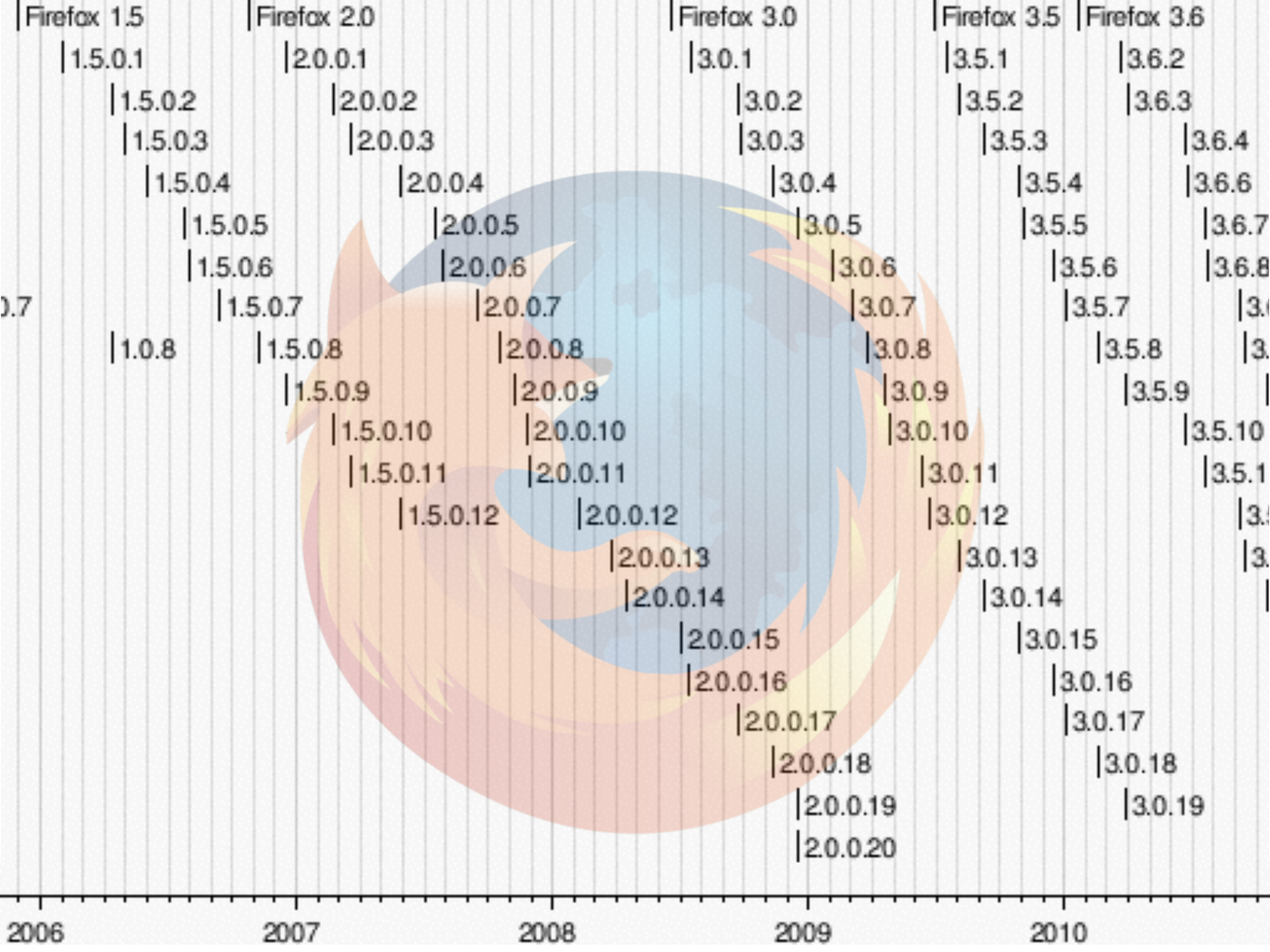


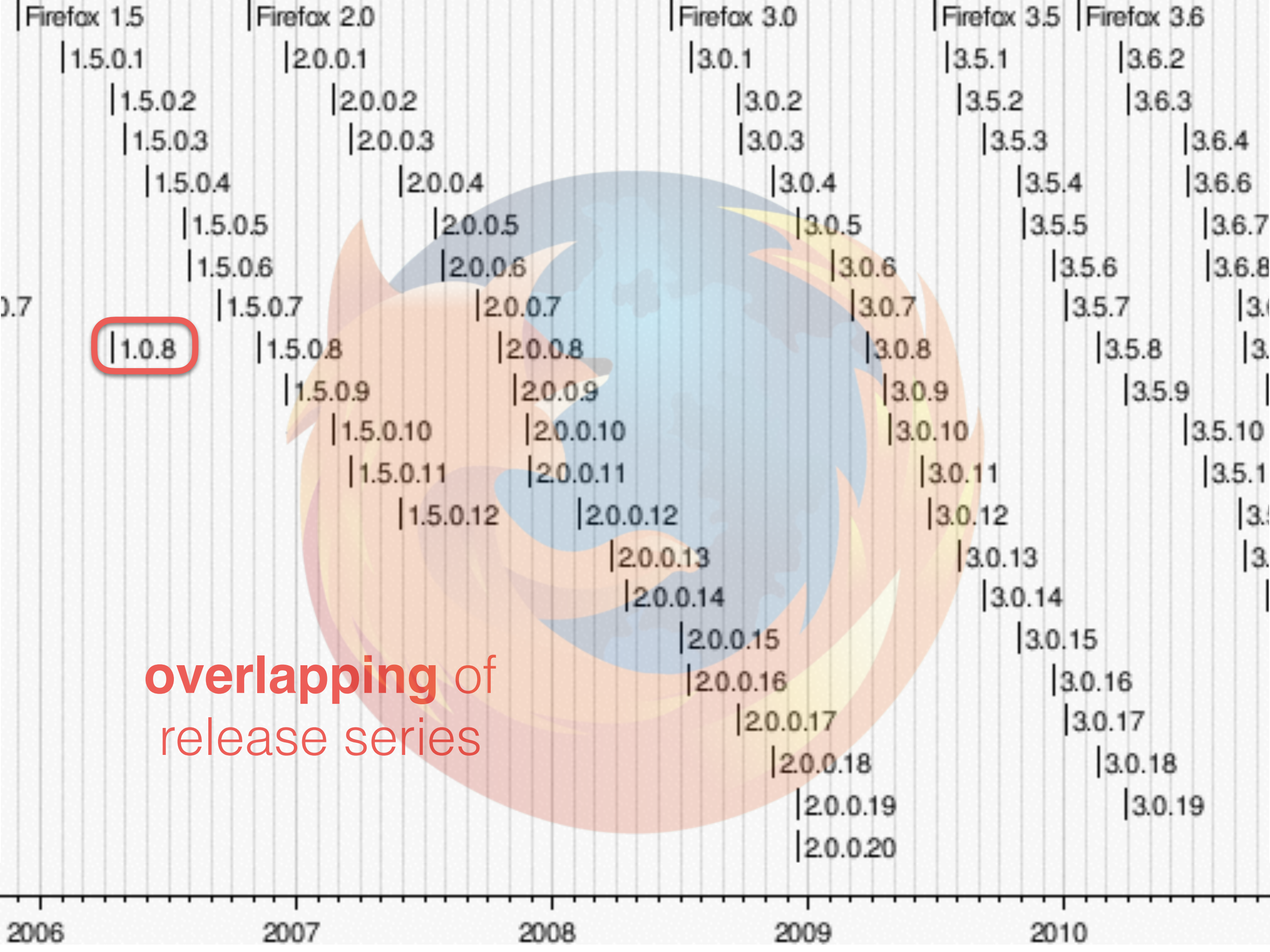
Features can be Older than the Cycle Time



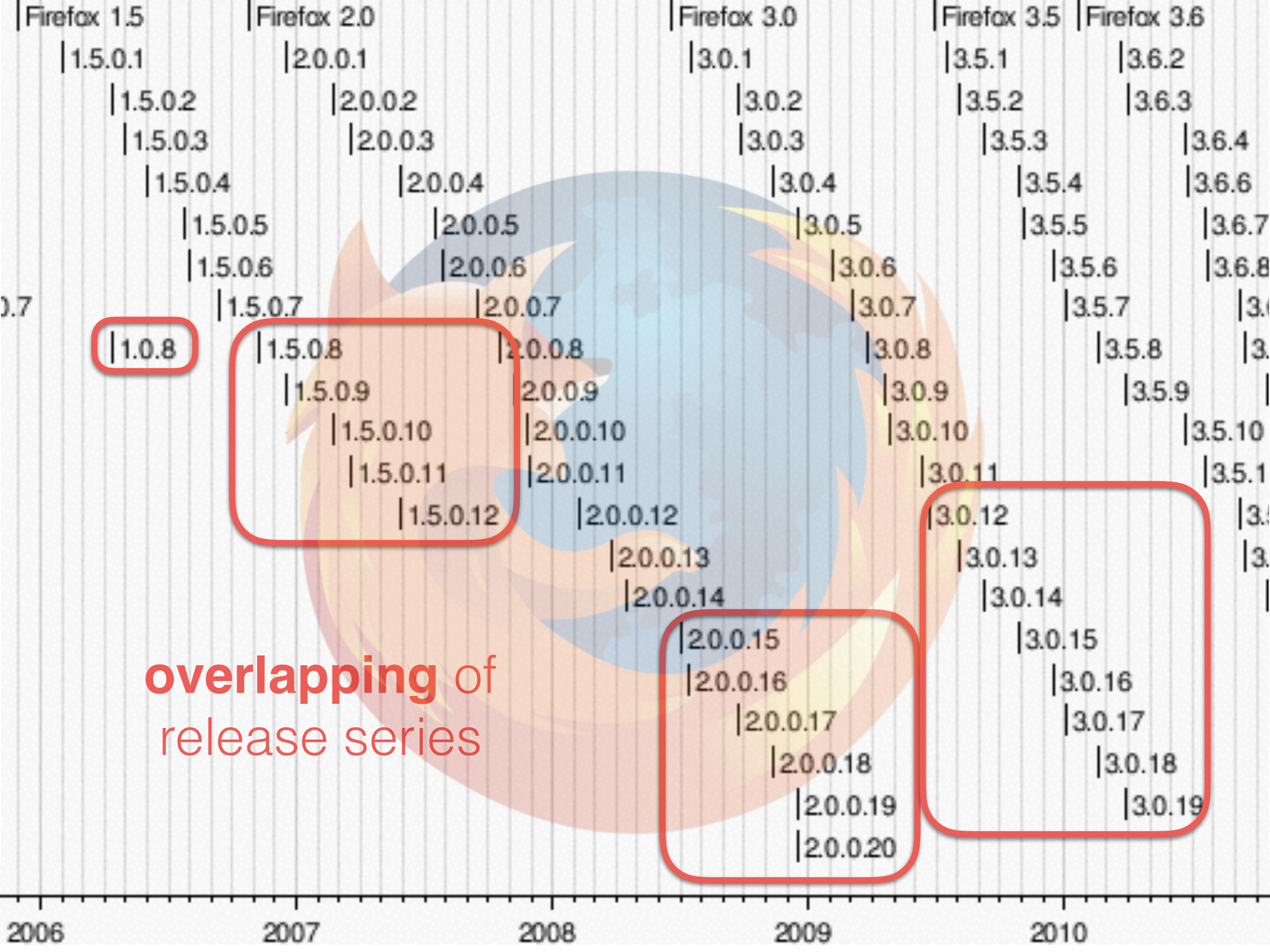
Features can be Older than the Cycle Time

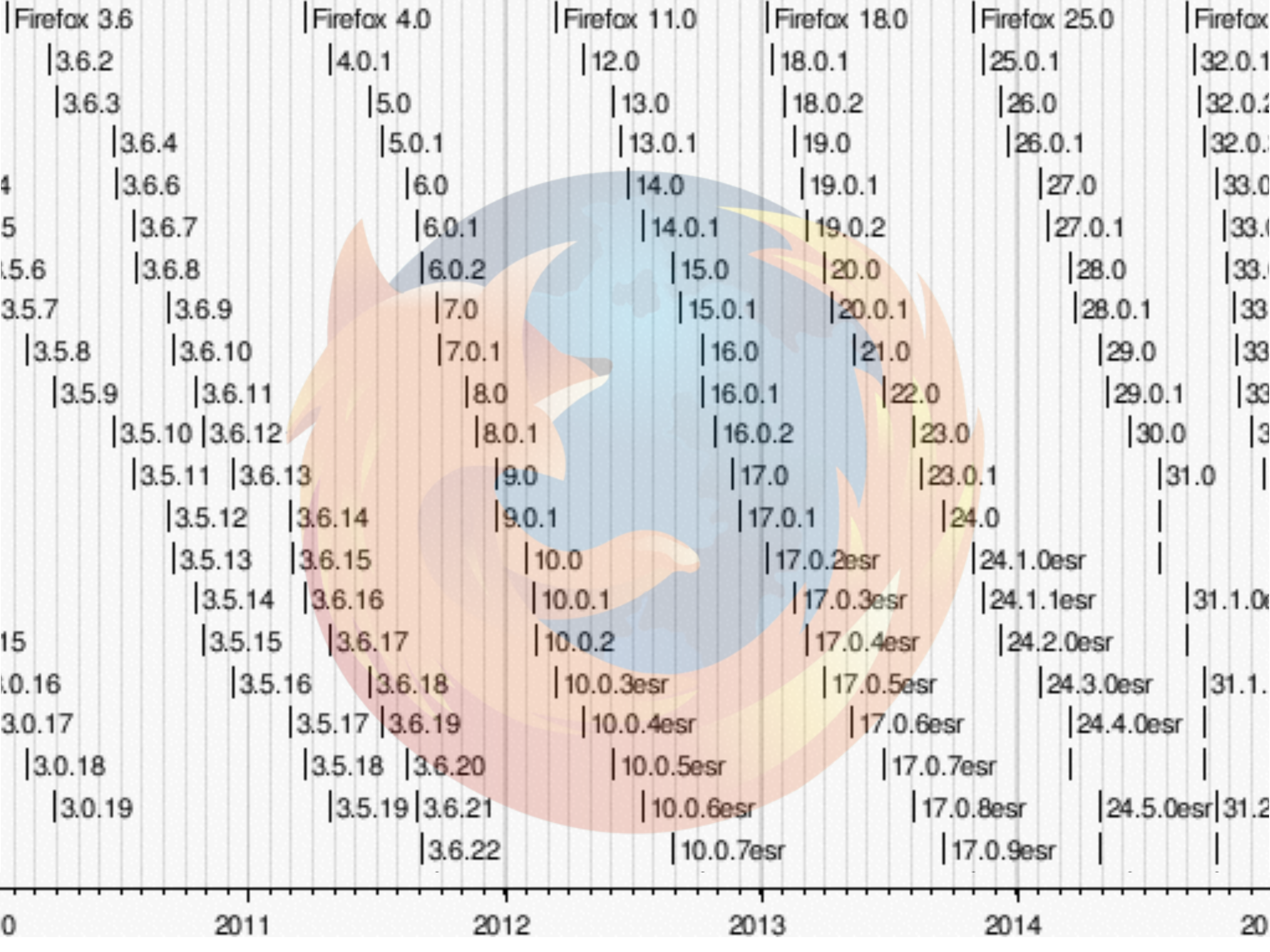


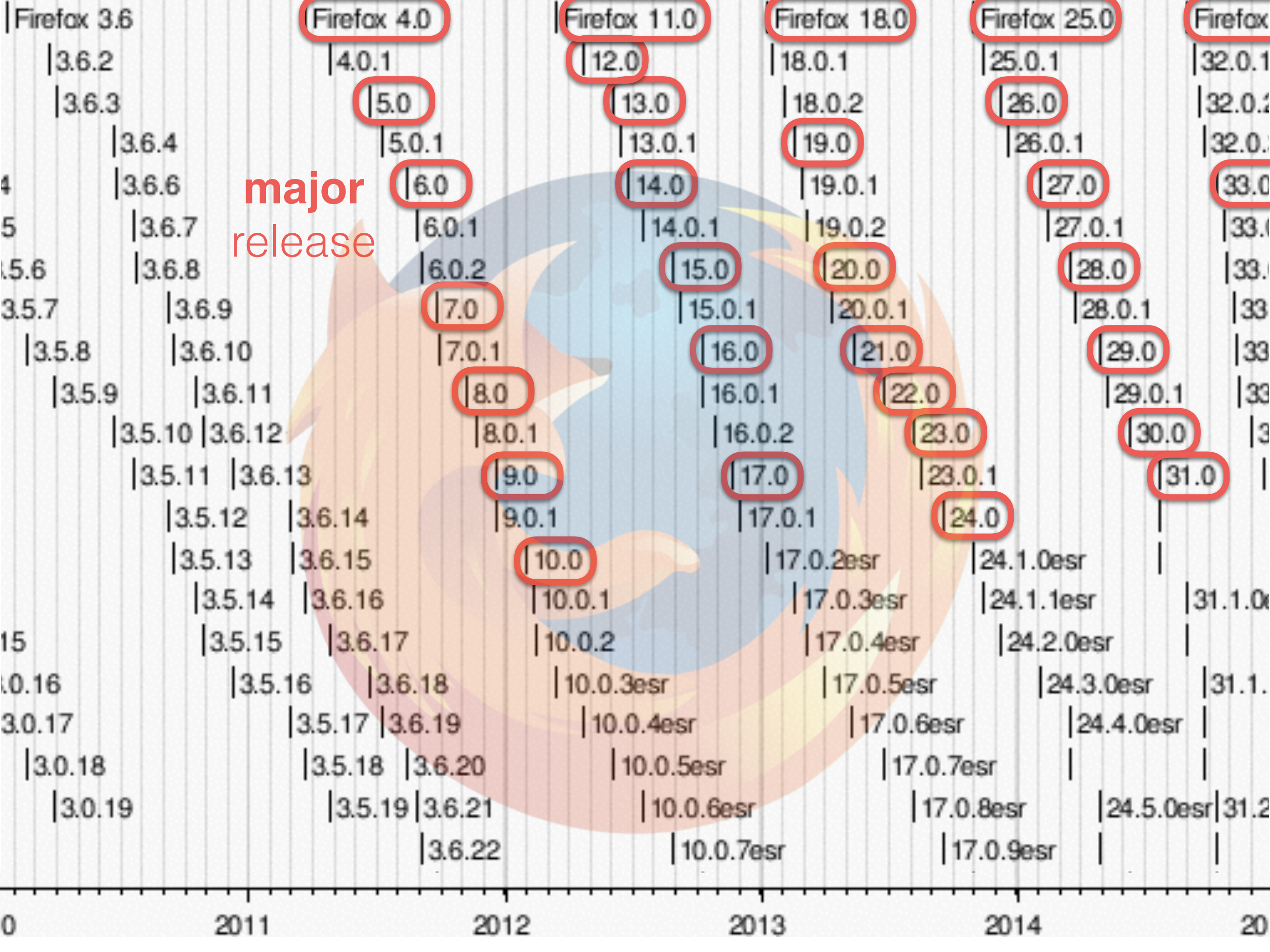


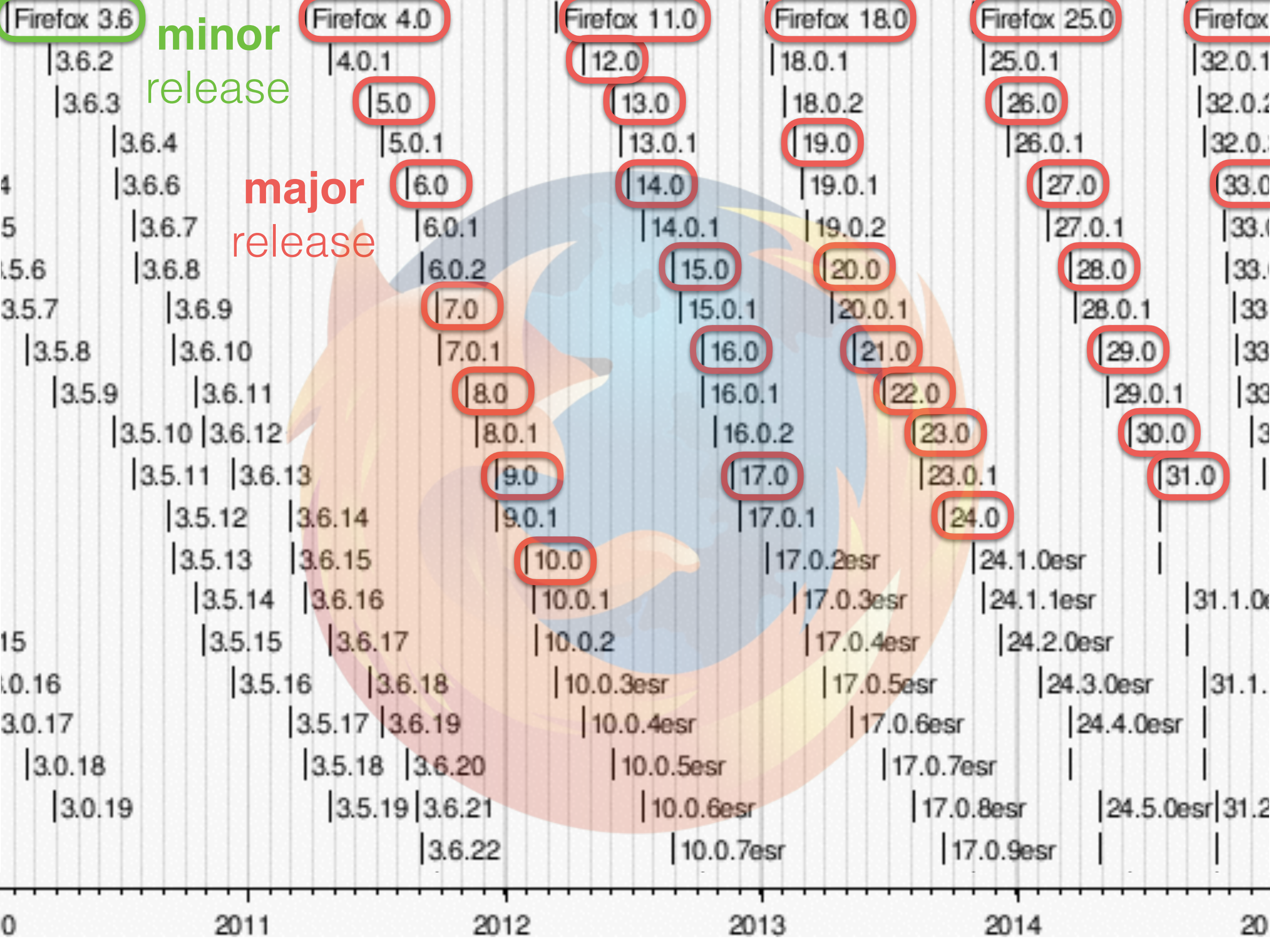


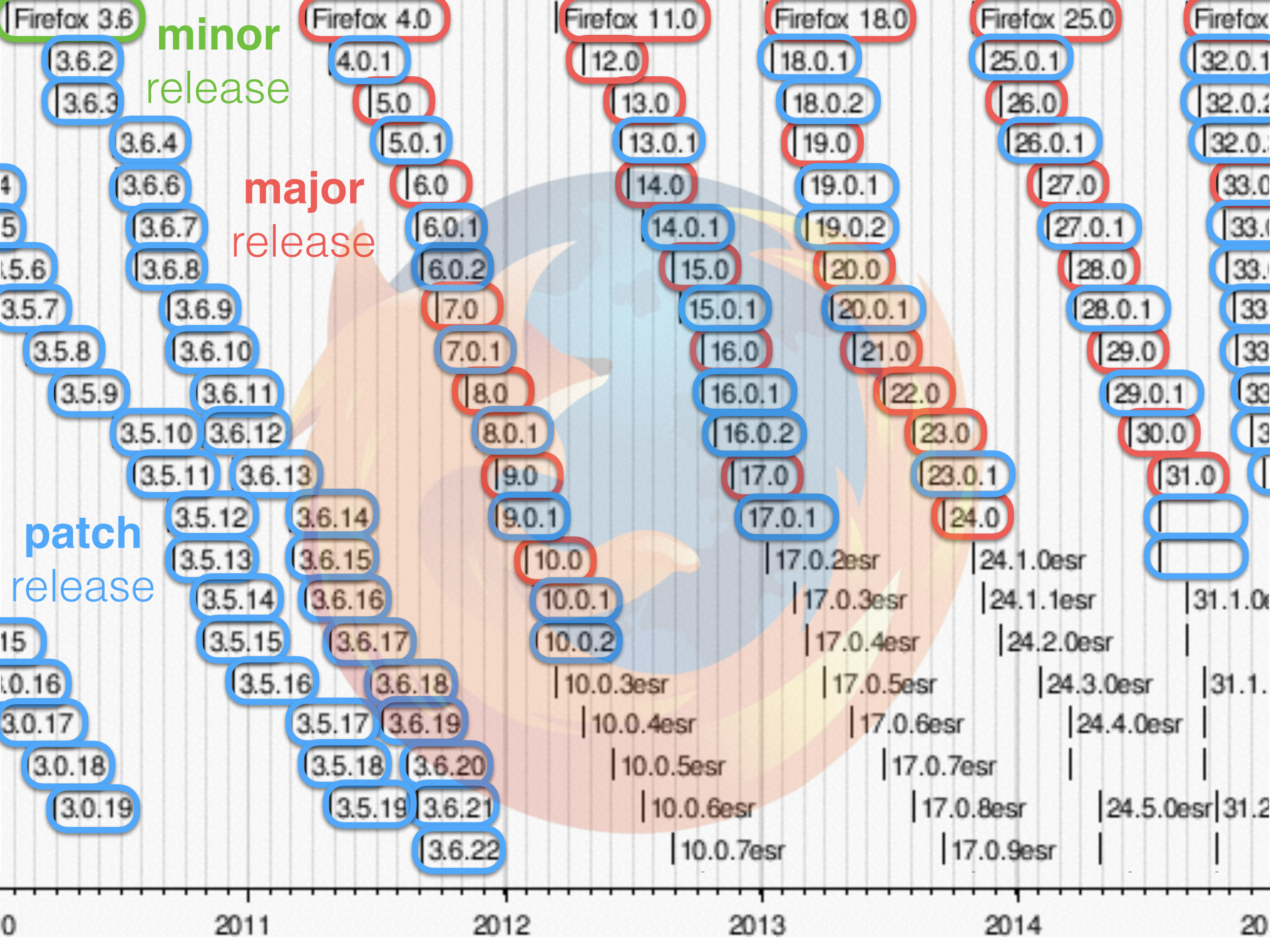
overlapping of
release series

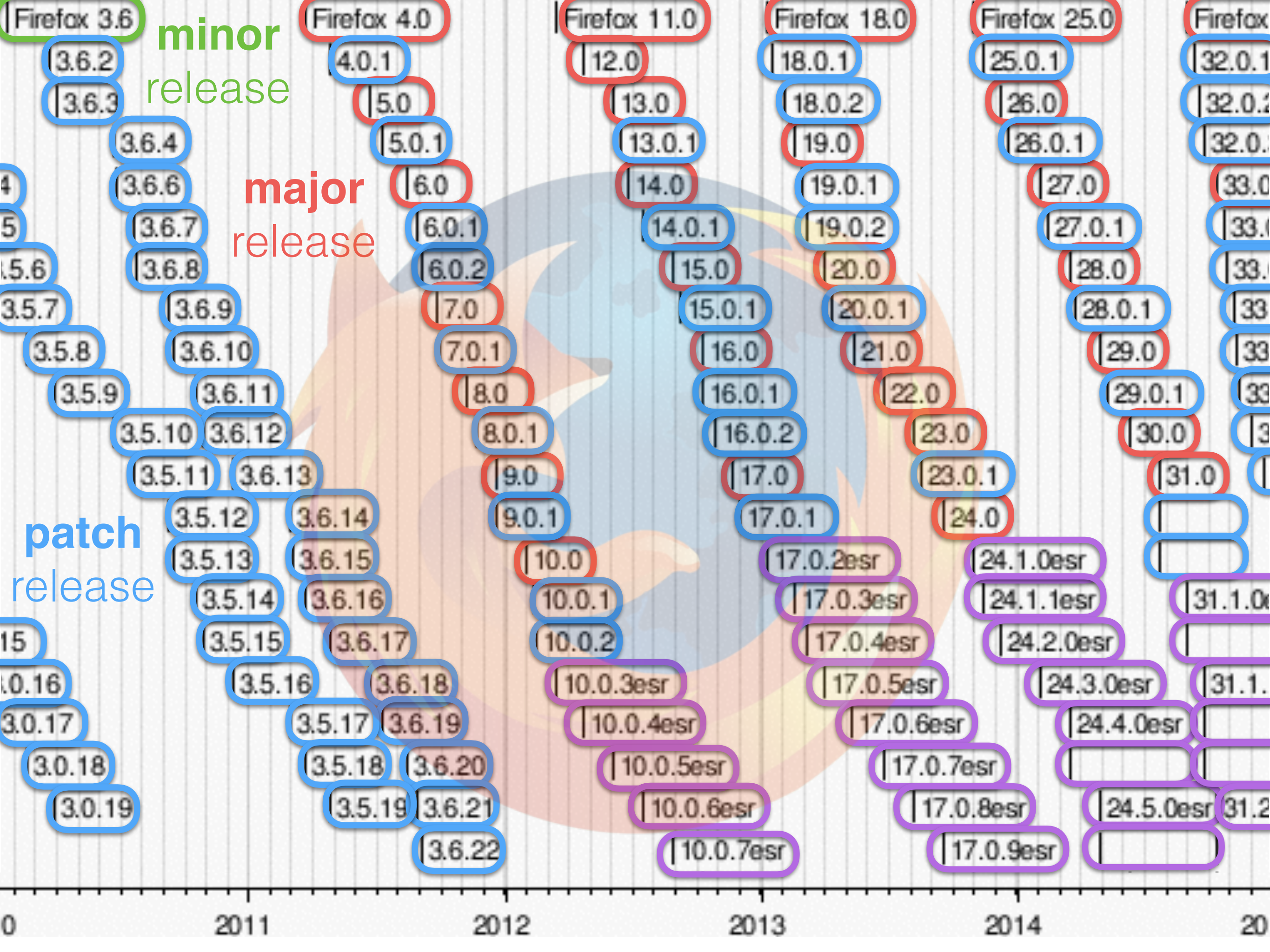


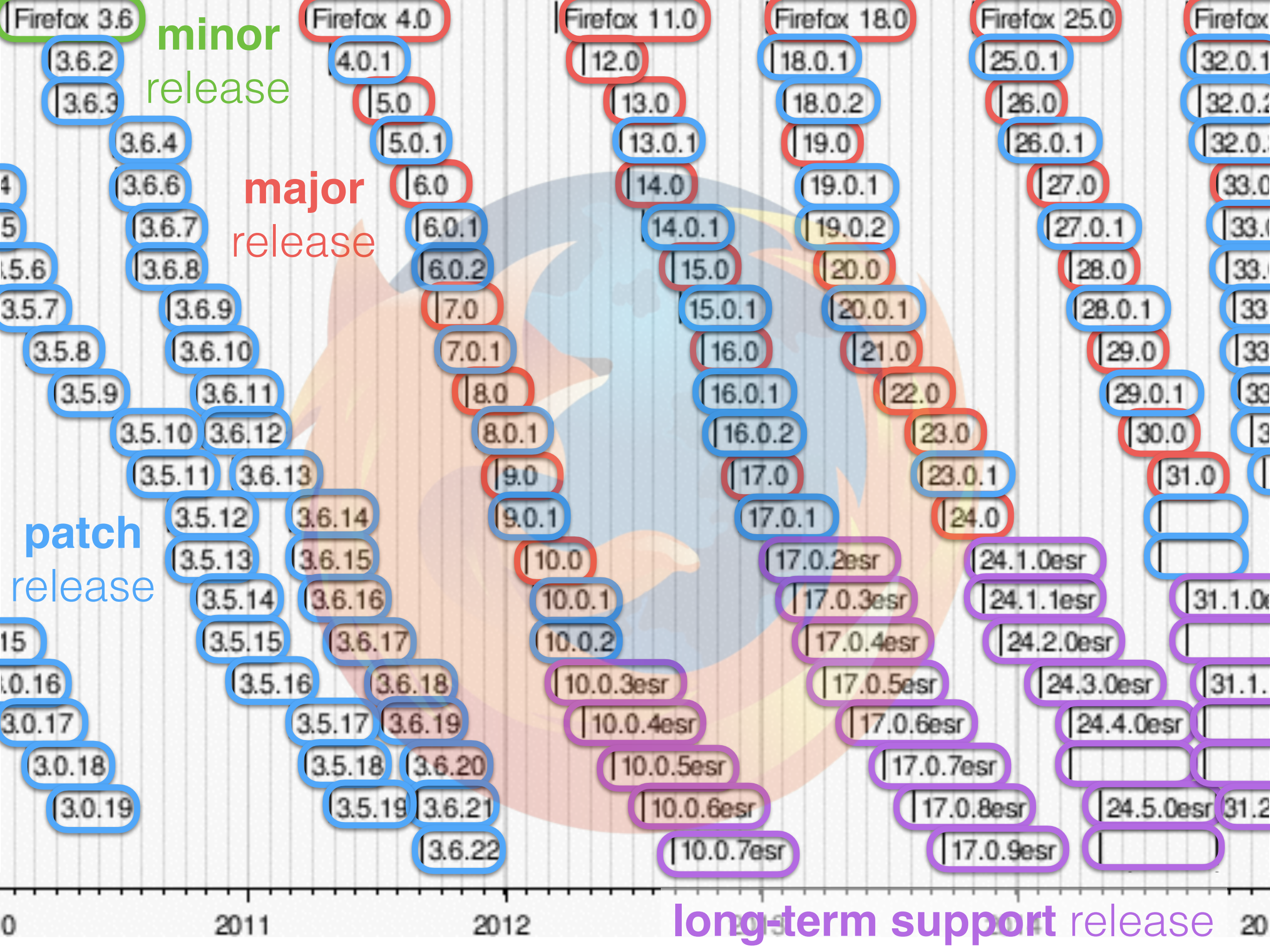














So, these students should
always check the cycle
time, type of release and
release overlap?



So, these students should
always check the cycle
time, type of release and
release overlap?

YES!





2. All Branches are Equal



~~2. All Branches are Equal~~



Weird... The size of this
project fluctuates between
50k and 45k lines!





Weird... The size of this project fluctuates between 50k and 45k lines!



Hmm, did you select the relevant branch? Several are **developed in parallel!**

Example: Analyzing Defects (1 Branch)



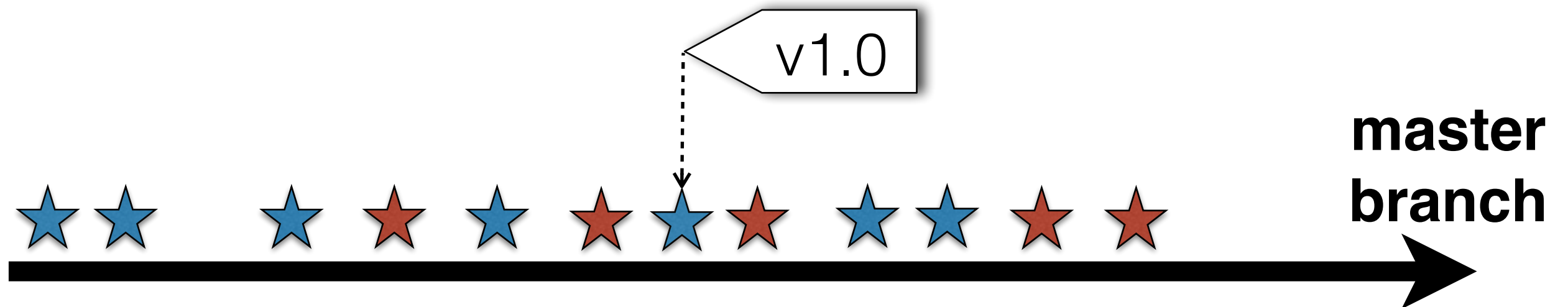
Commit types:
★ Feature ★ Fix ★ Merge

Example: Analyzing Defects (1 Branch)



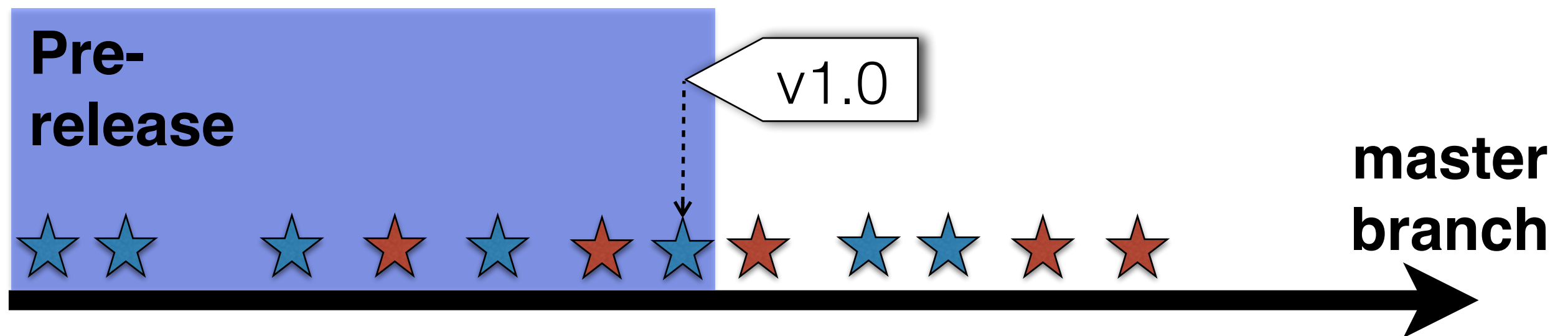
Commit types:
★ Feature ★ Fix ★ Merge

Example: Analyzing Defects (1 Branch)



Commit types:
★ Feature ★ Fix ★ Merge

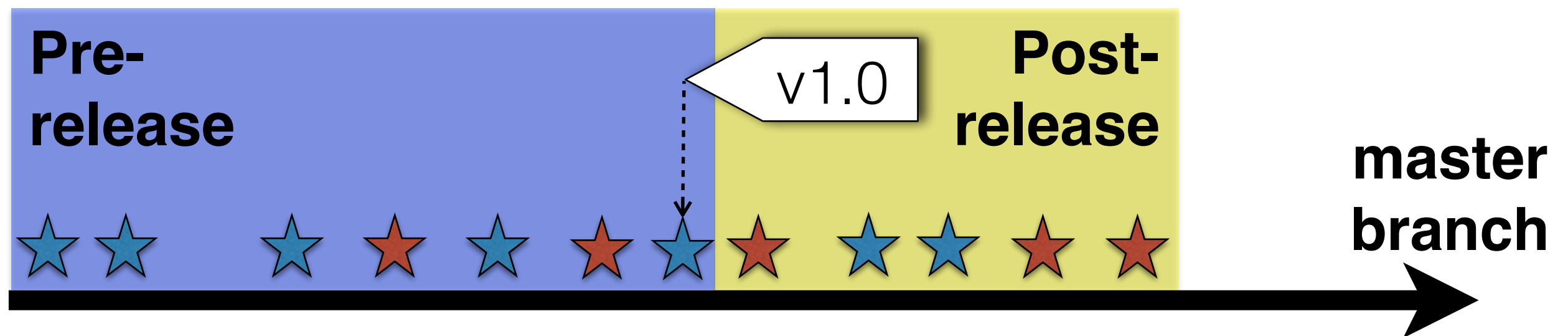
Example: Analyzing Defects (1 Branch)



Commit types:

★ Feature ★ Fix ★ Merge

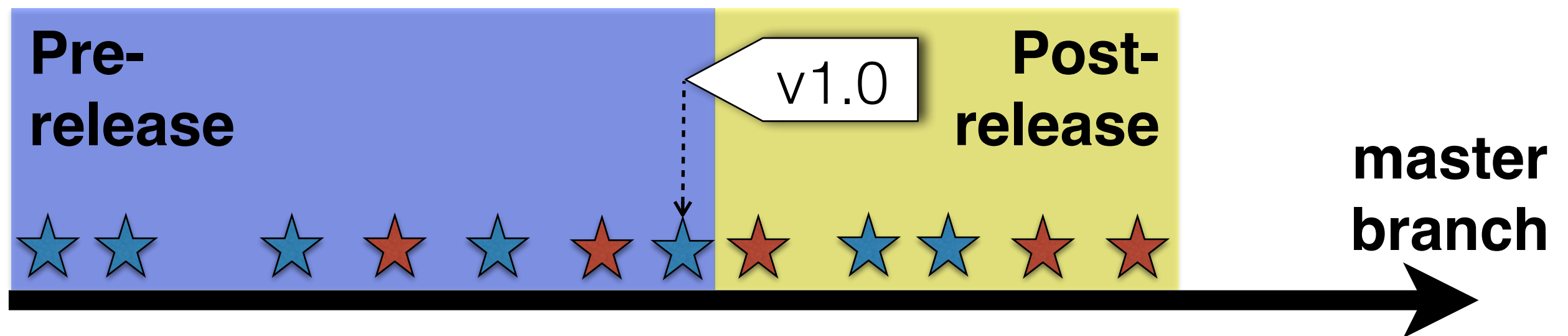
Example: Analyzing Defects (1 Branch)



Commit types:

★ Feature ★ Fix ★ Merge

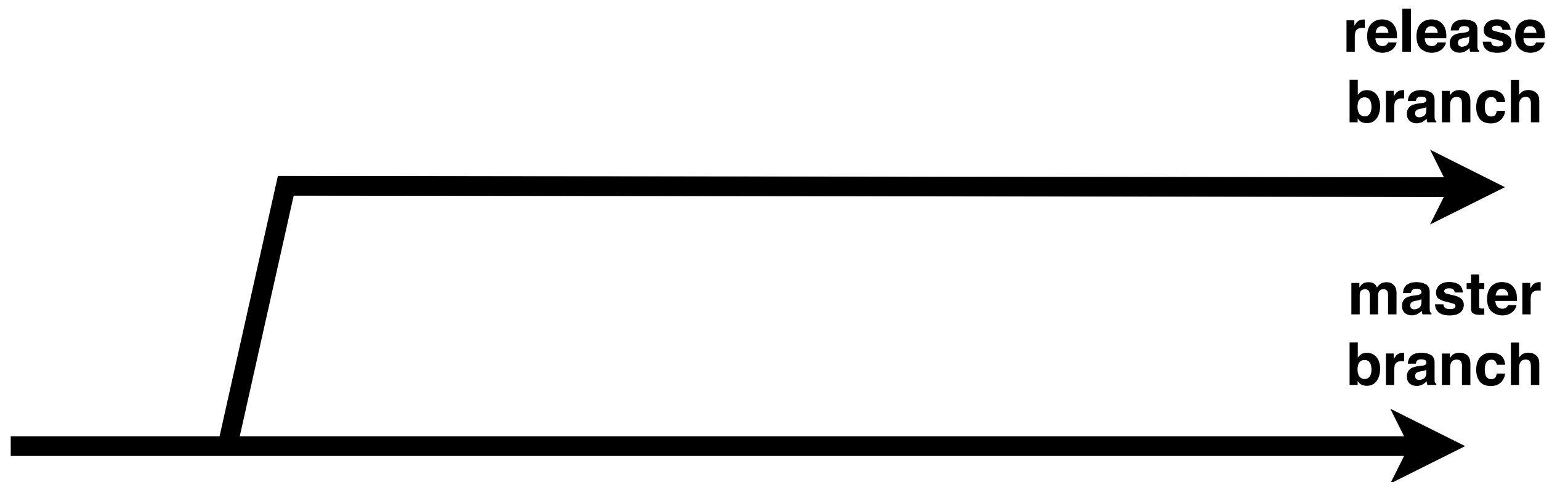
Example: Analyzing Defects (1 Branch)



Commit types:

★ Feature ★ Fix ★ Merge

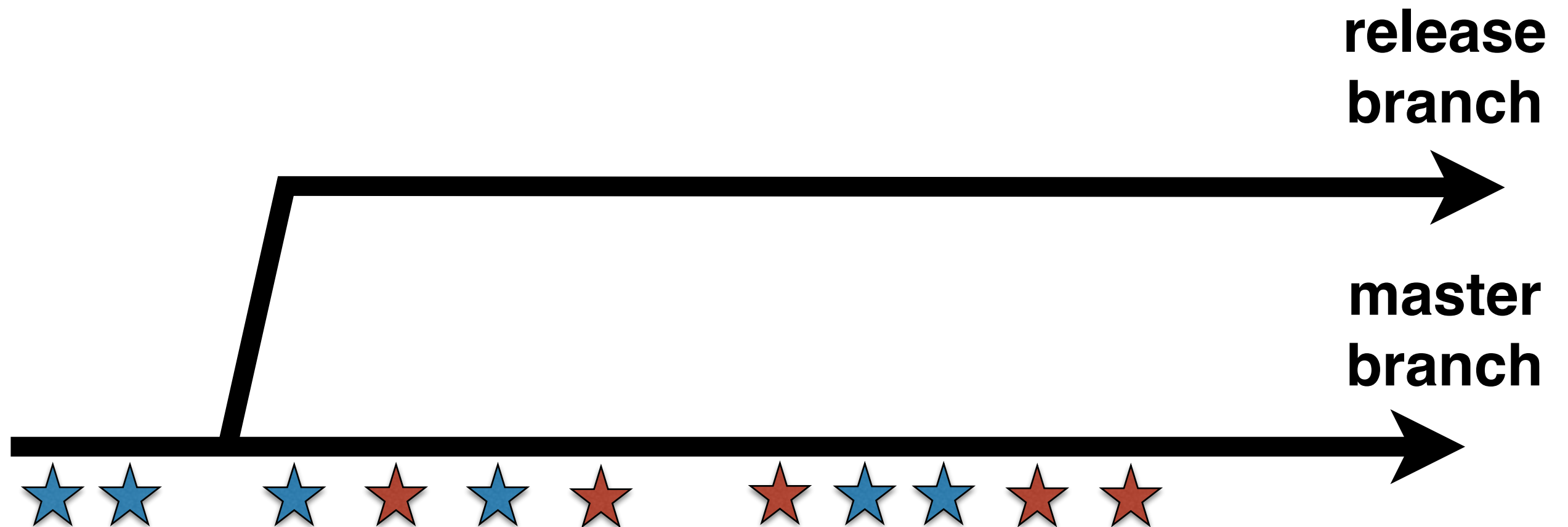
Example: Analyzing Defects (>1 Branch)



Commit types:

★ Feature ★ Fix ★ Merge

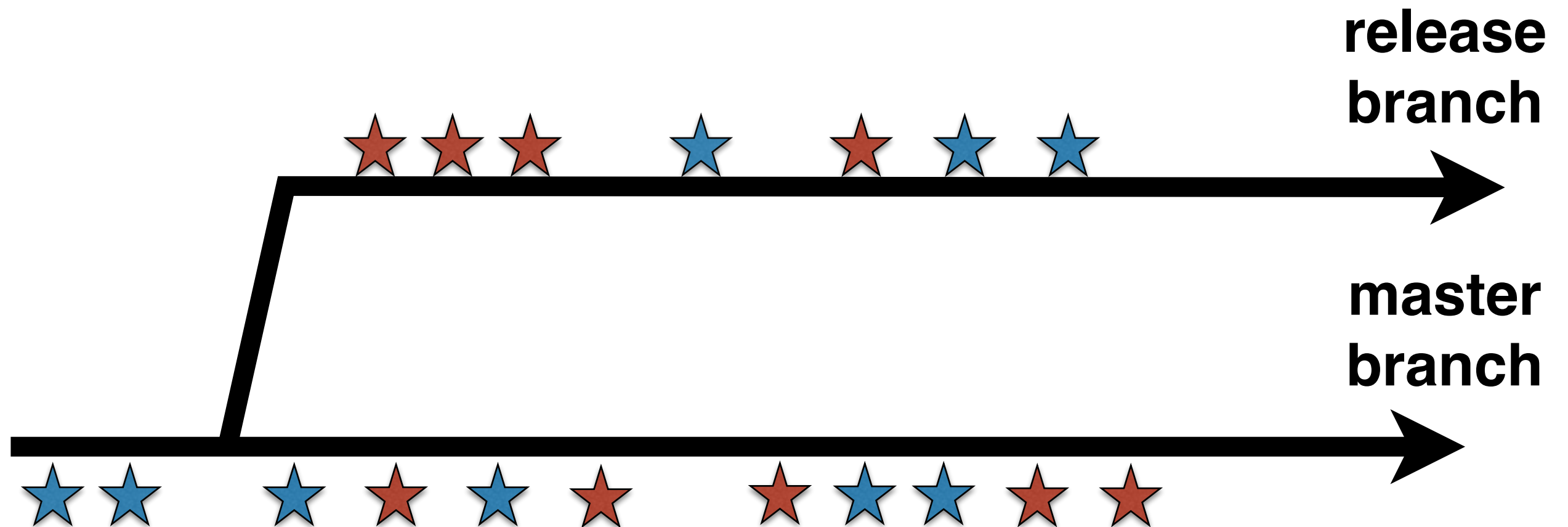
Example: Analyzing Defects (>1 Branch)



Commit types:

★ Feature ★ Fix ★ Merge

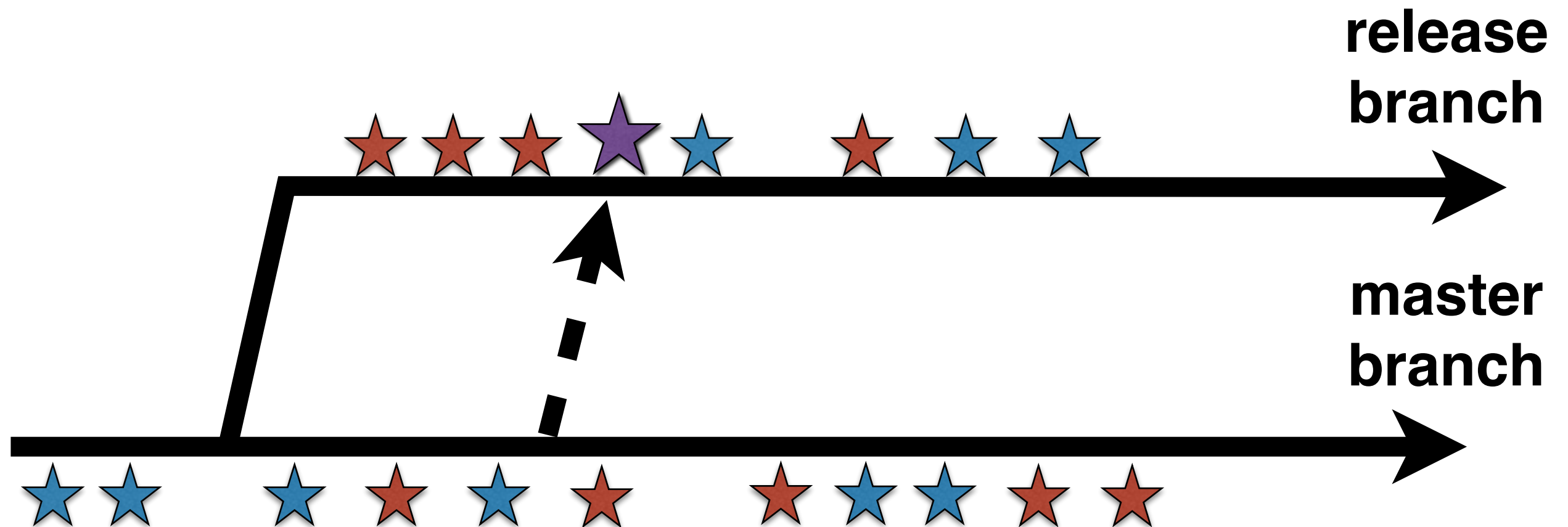
Example: Analyzing Defects (>1 Branch)



Commit types:

★ Feature ★ Fix ★ Merge

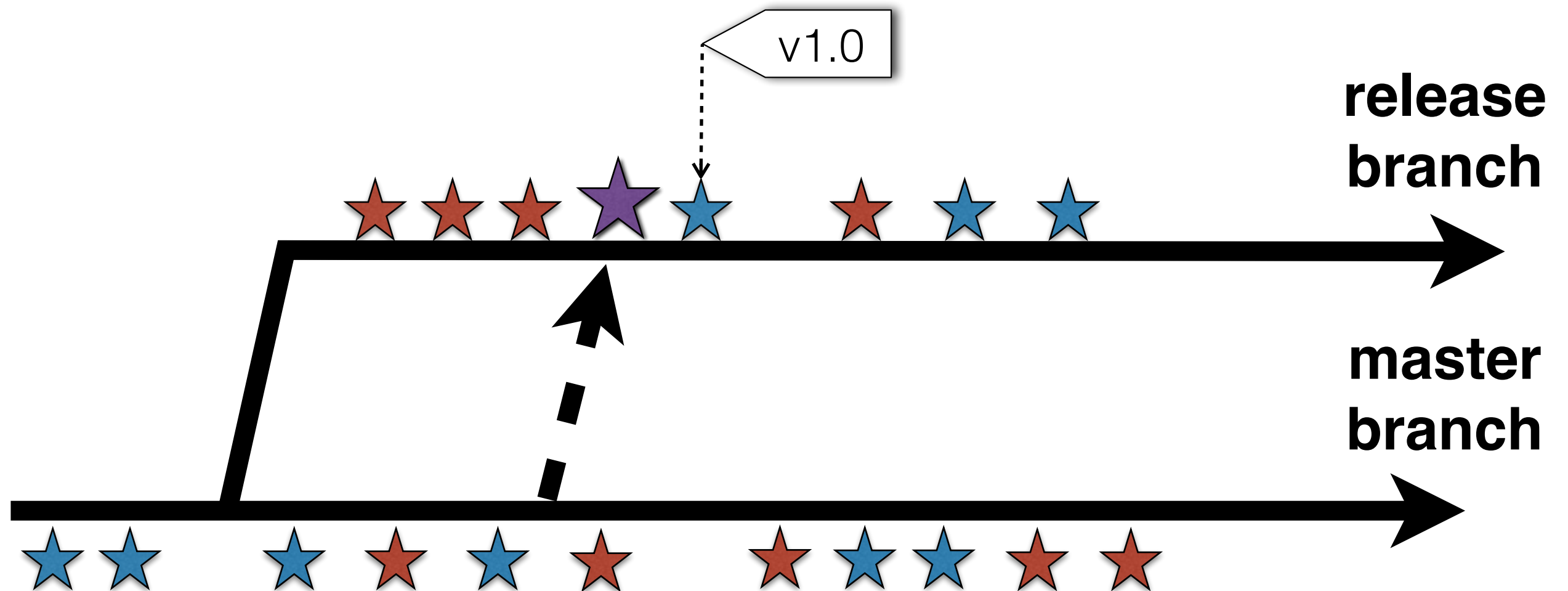
Example: Analyzing Defects (>1 Branch)



Commit types:

★ Feature ★ Fix ★ Merge

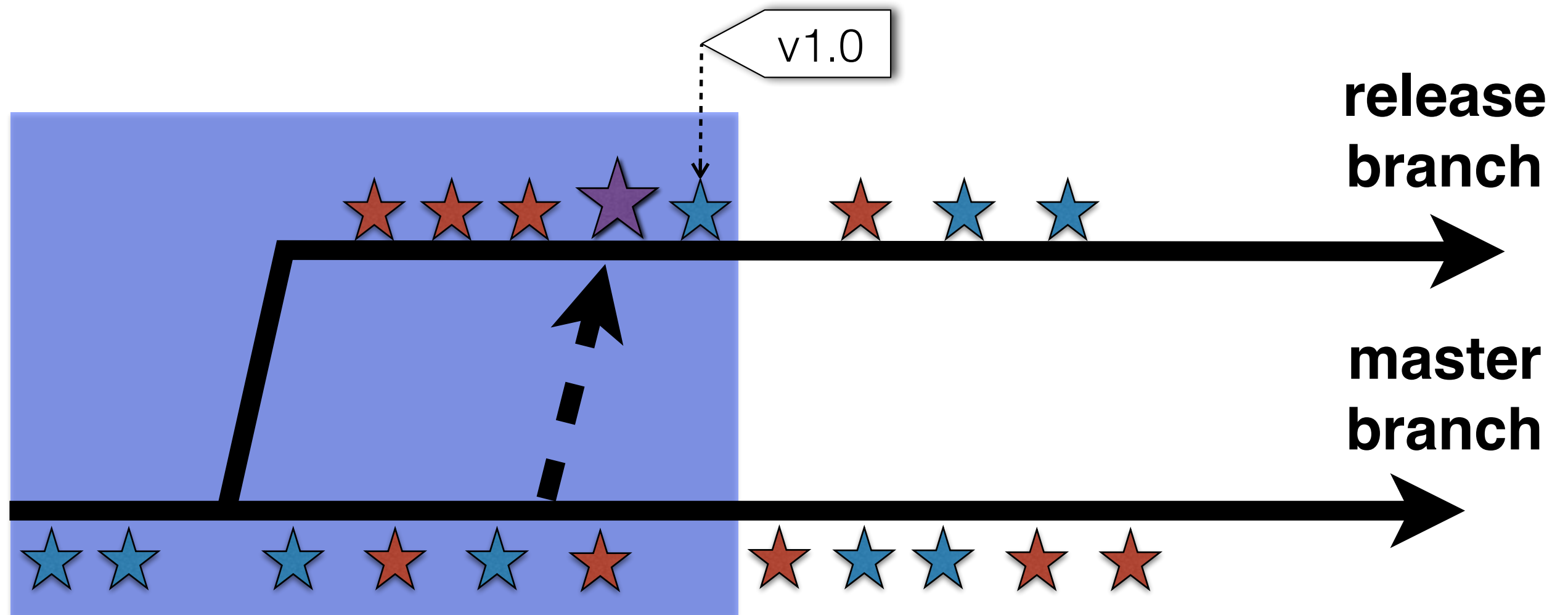
Example: Analyzing Defects (>1 Branch)



Commit types:

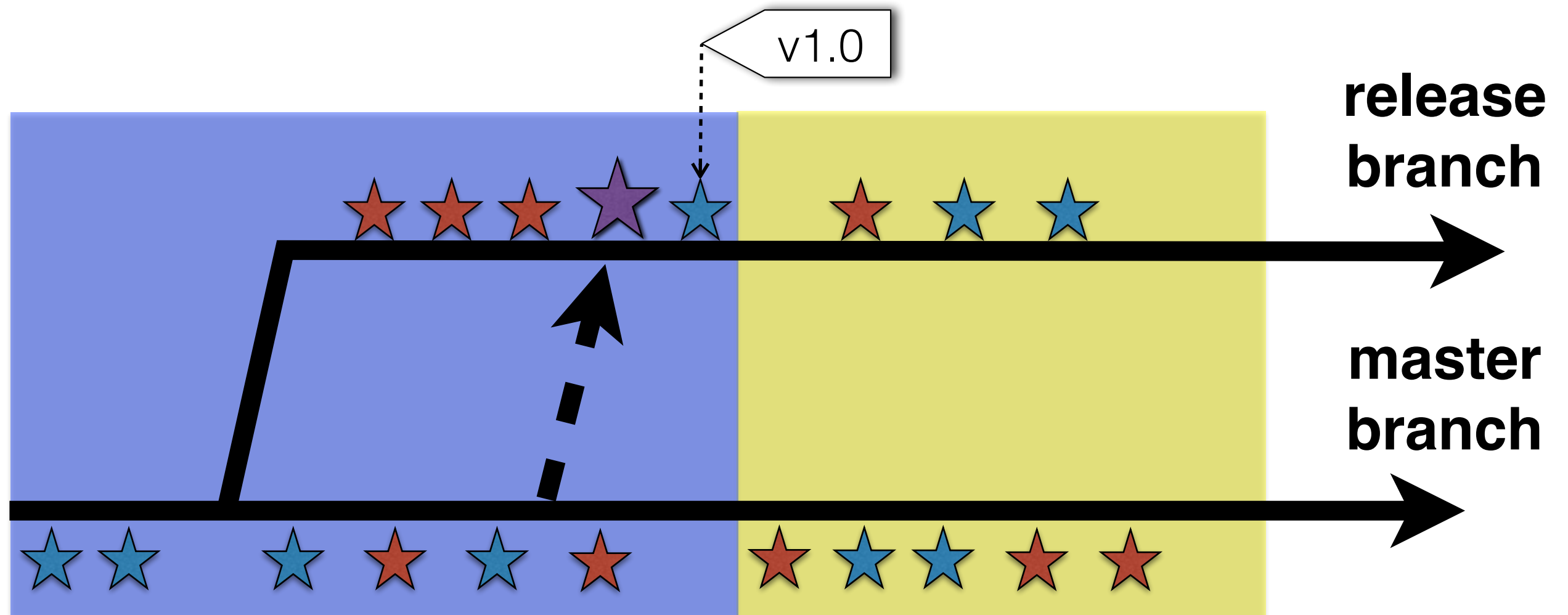
★ Feature ★ Fix ★ Merge

Example: Analyzing Defects (>1 Branch)



Commit types:
★ Feature ★ Fix ★ Merge

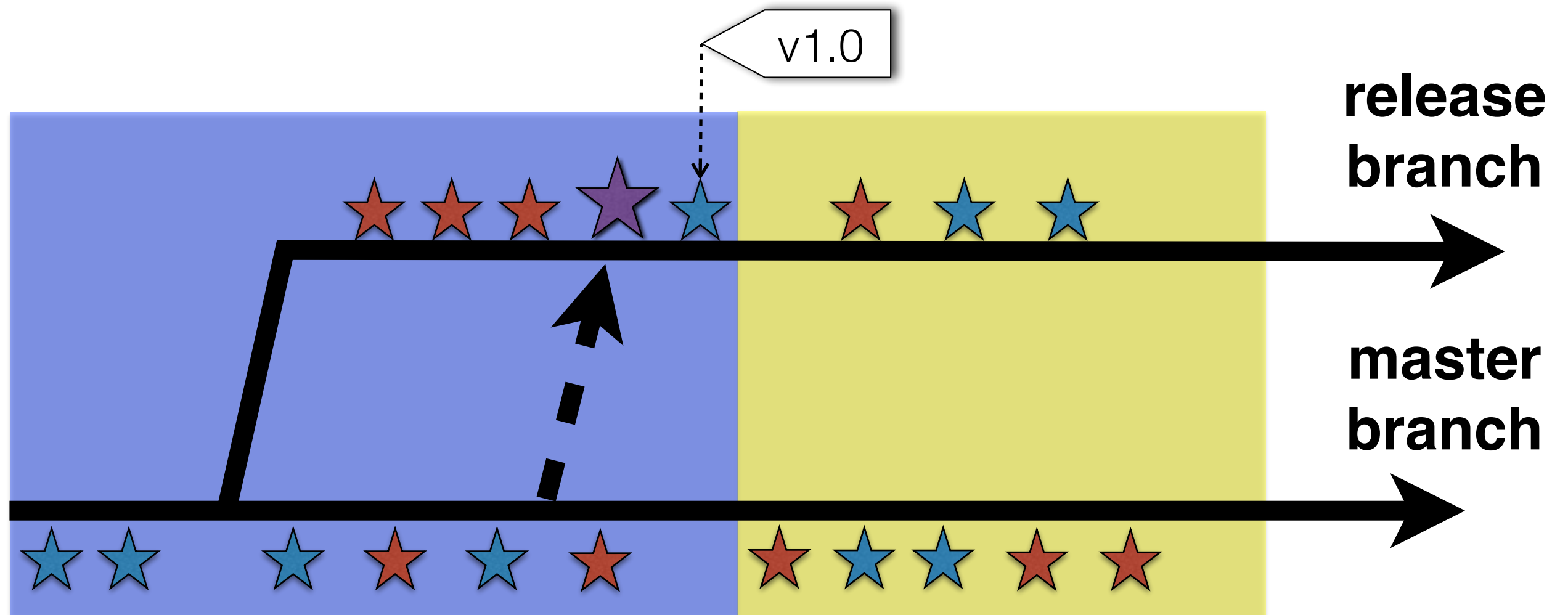
Example: Analyzing Defects (>1 Branch)



Commit types:

★ Feature ★ Fix ★ Merge

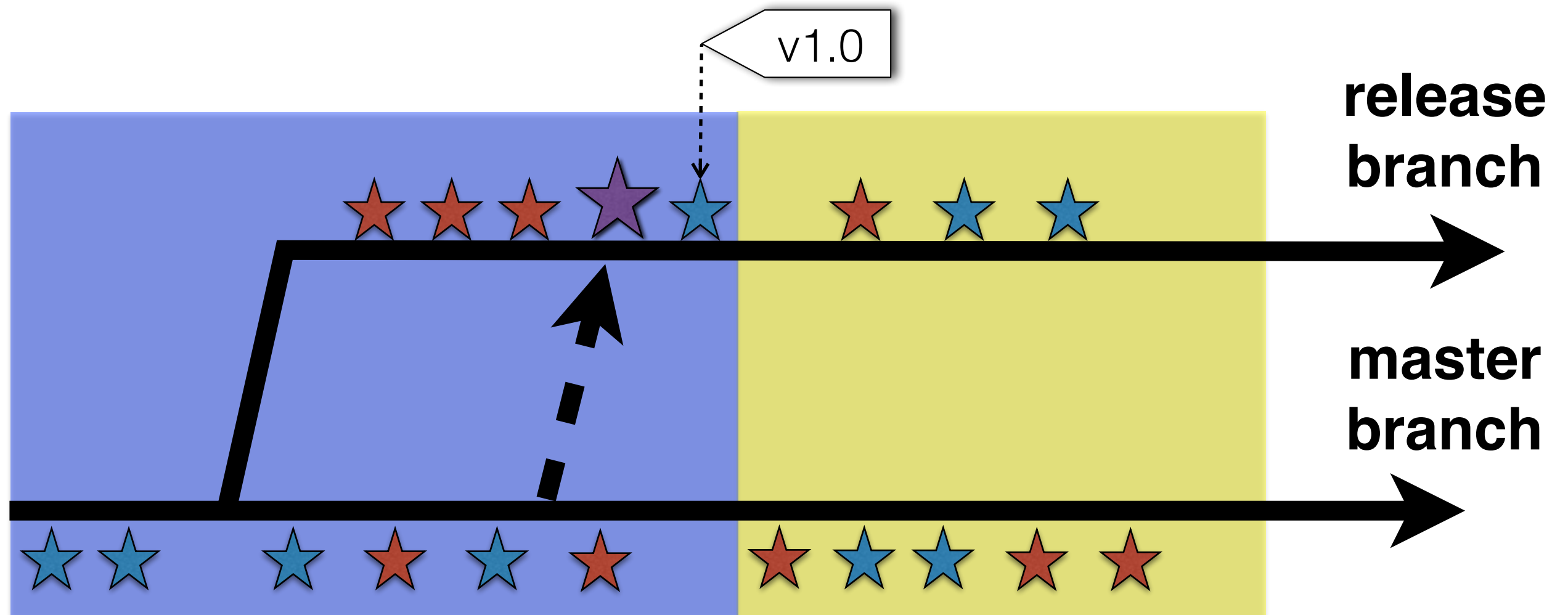
Example: Analyzing Defects (>1 Branch)



Commit types:

★ Feature ★ Fix ★ Merge

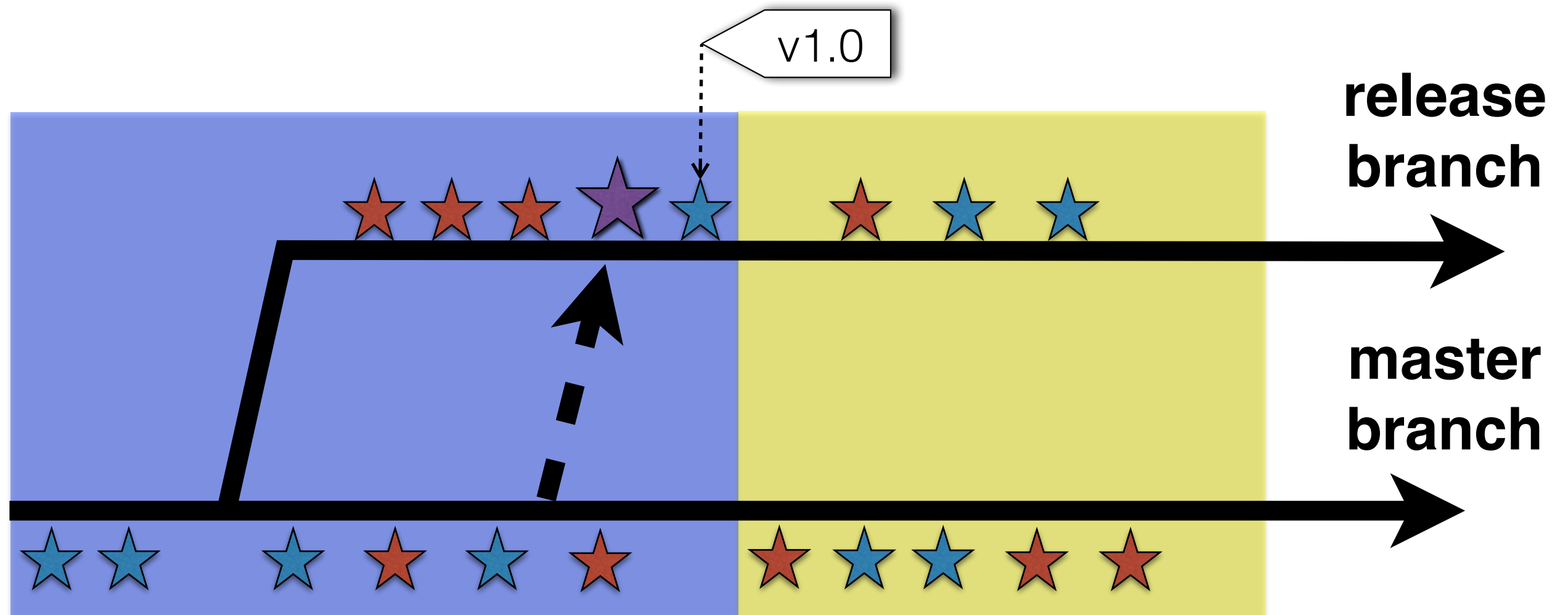
Example: Analyzing Defects (>1 Branch)



Commit types:

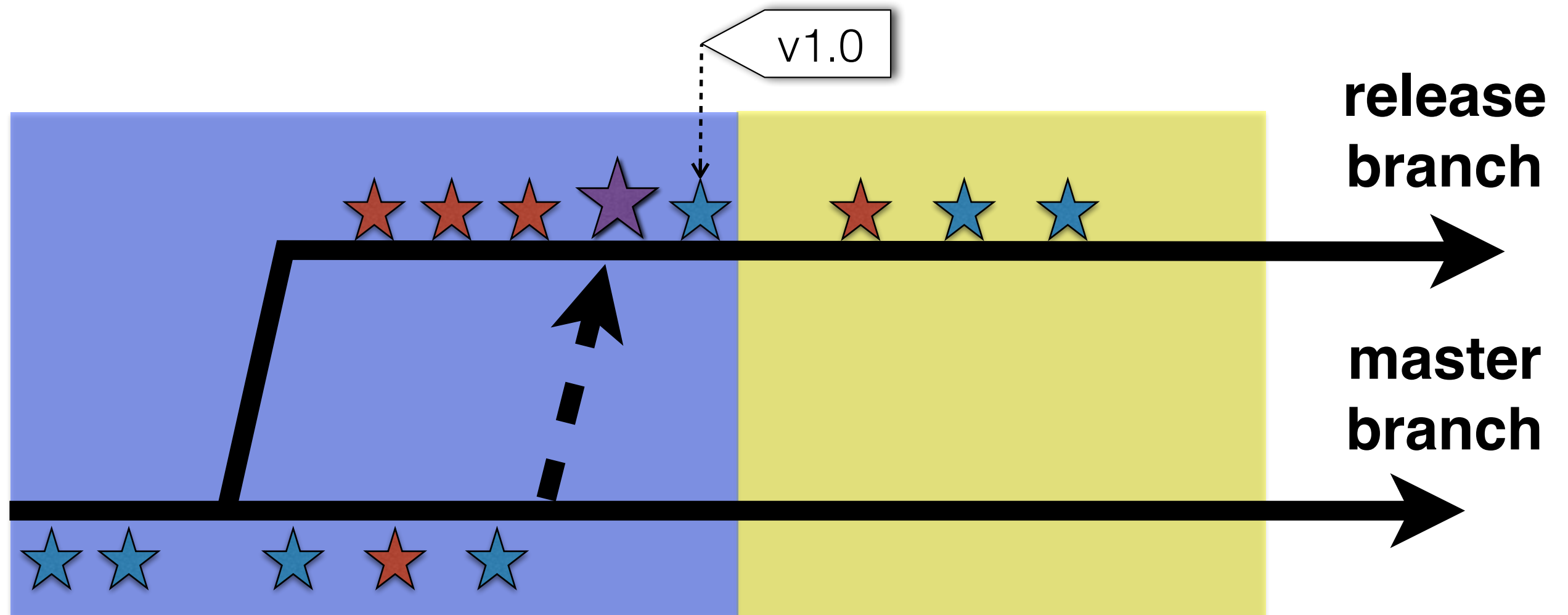
★ Feature ★ Fix ★ Merge

Example: Analyzing Defects (>1 Branch)



Commit types:
★ Feature ★ Fix ★ Merge

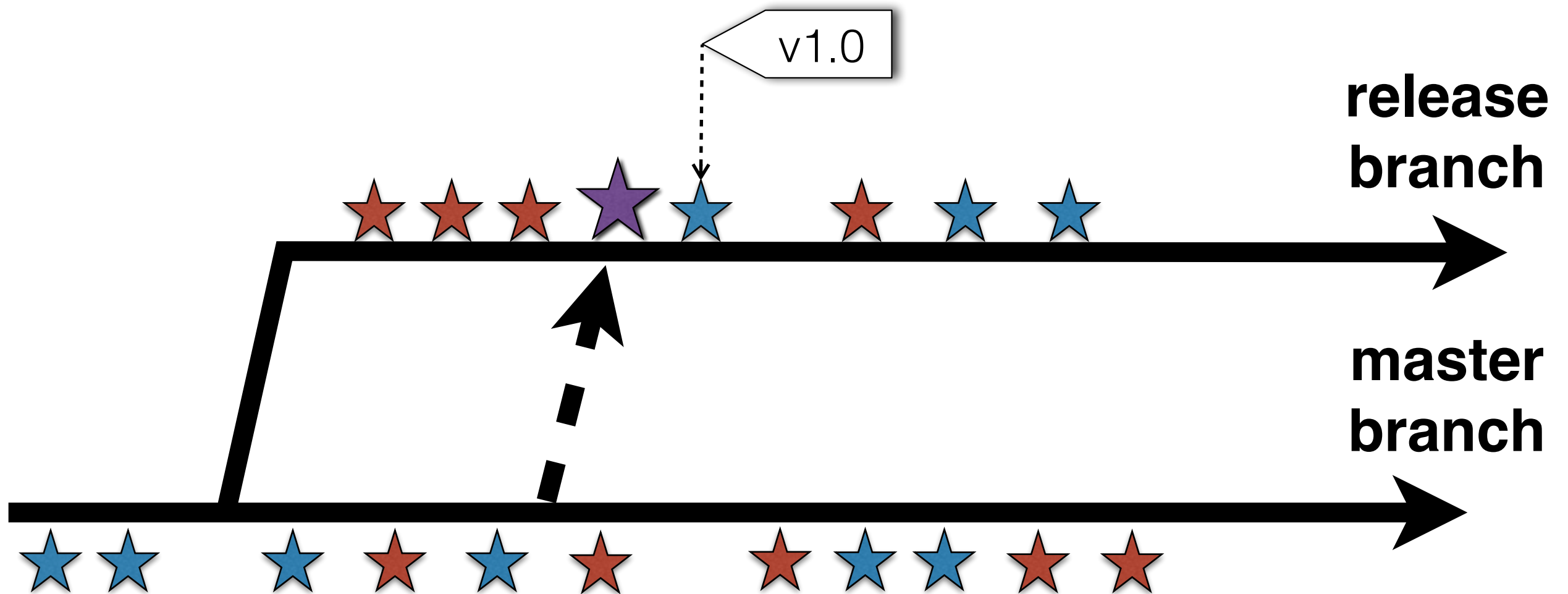
Example: Analyzing Defects (>1 Branch)



Commit types:

★ Feature ★ Fix ★ Merge

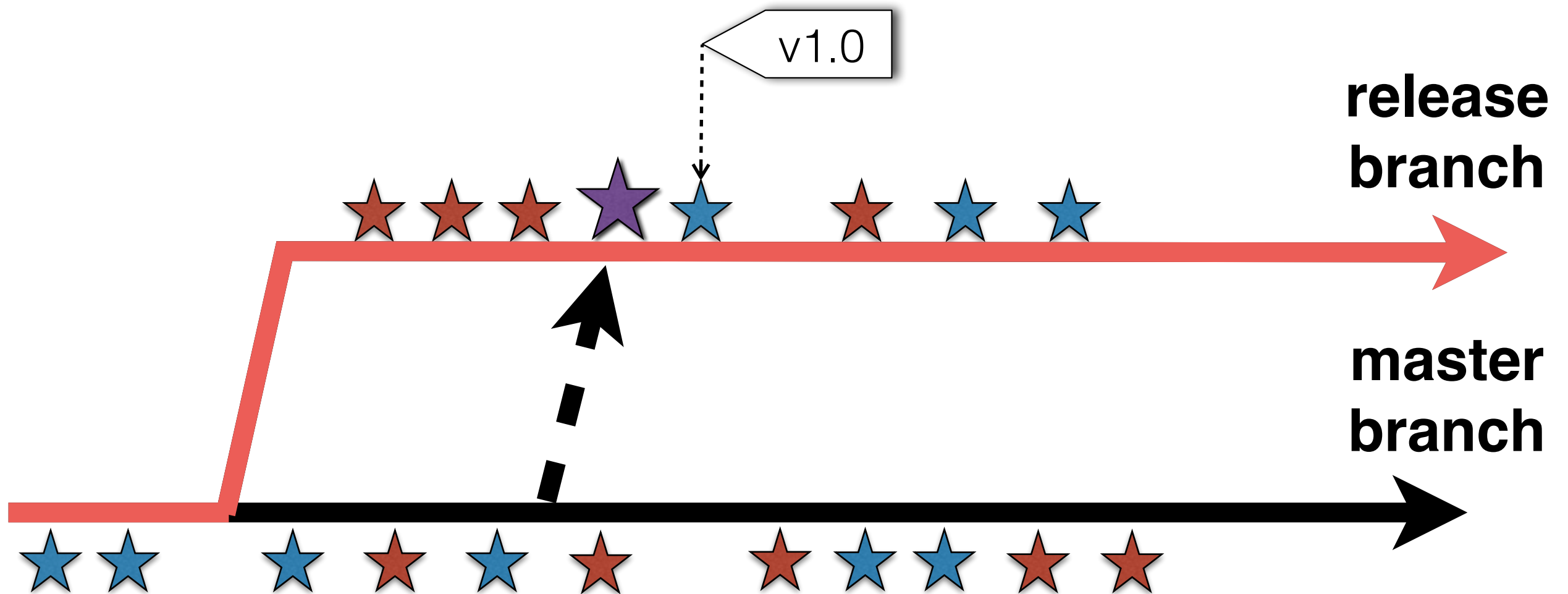
Solution: Select 1 Branch for Analysis and Expand Merge Commits



Commit types:

★ Feature ★ Fix ★ Merge

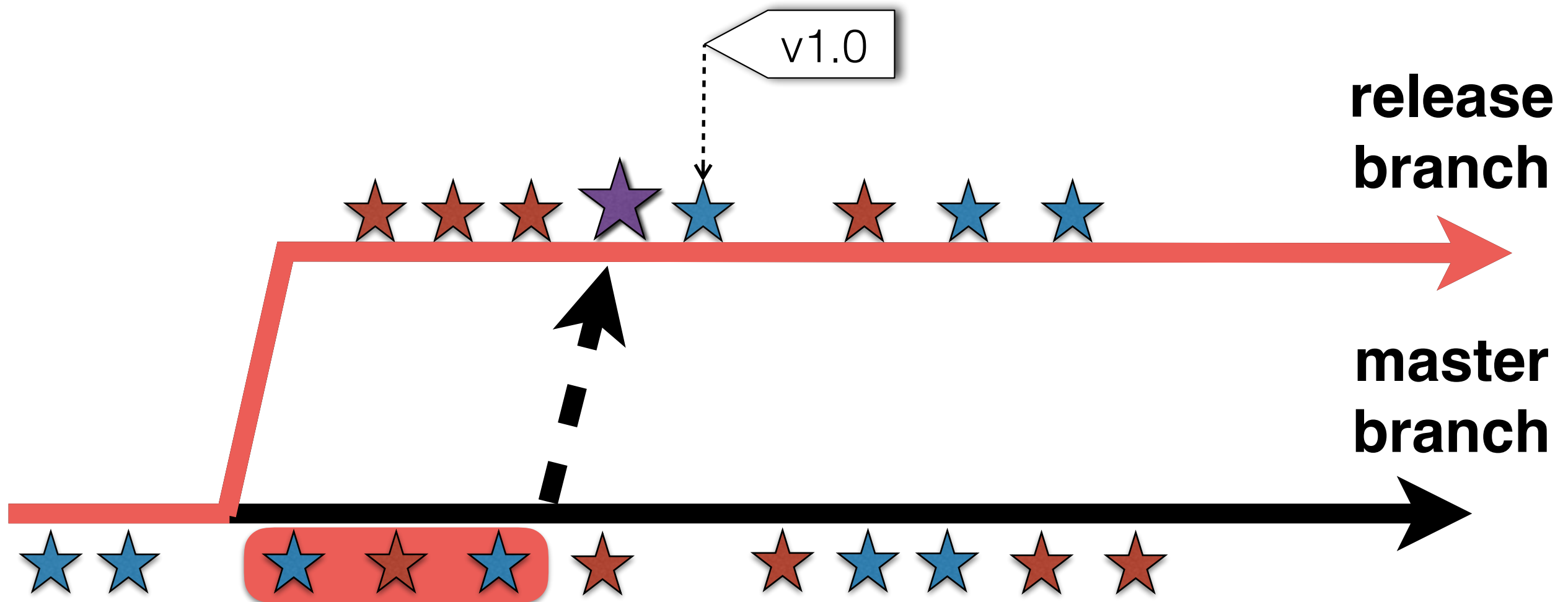
Solution: Select 1 Branch for Analysis and Expand Merge Commits



Commit types:

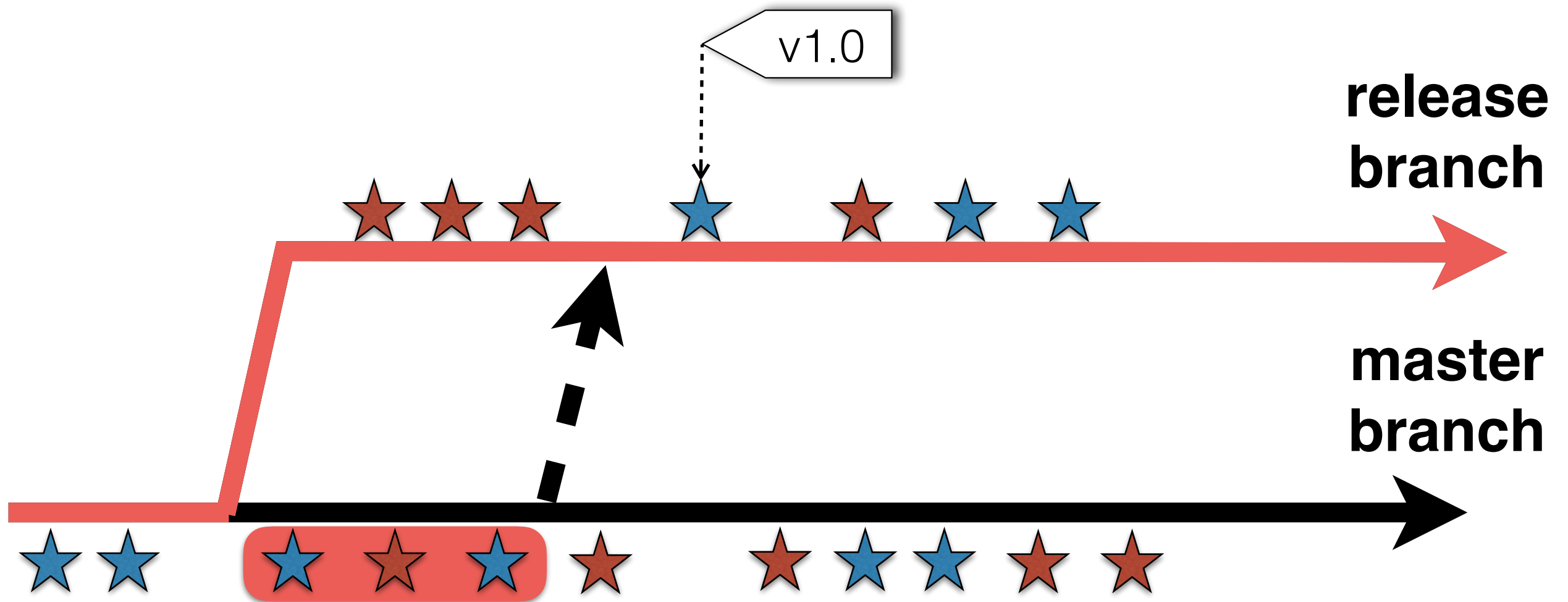
★ Feature ★ Fix ★ Merge

Solution: Select 1 Branch for Analysis and Expand Merge Commits



Commit types:
★ Feature ★ Fix ★ Merge

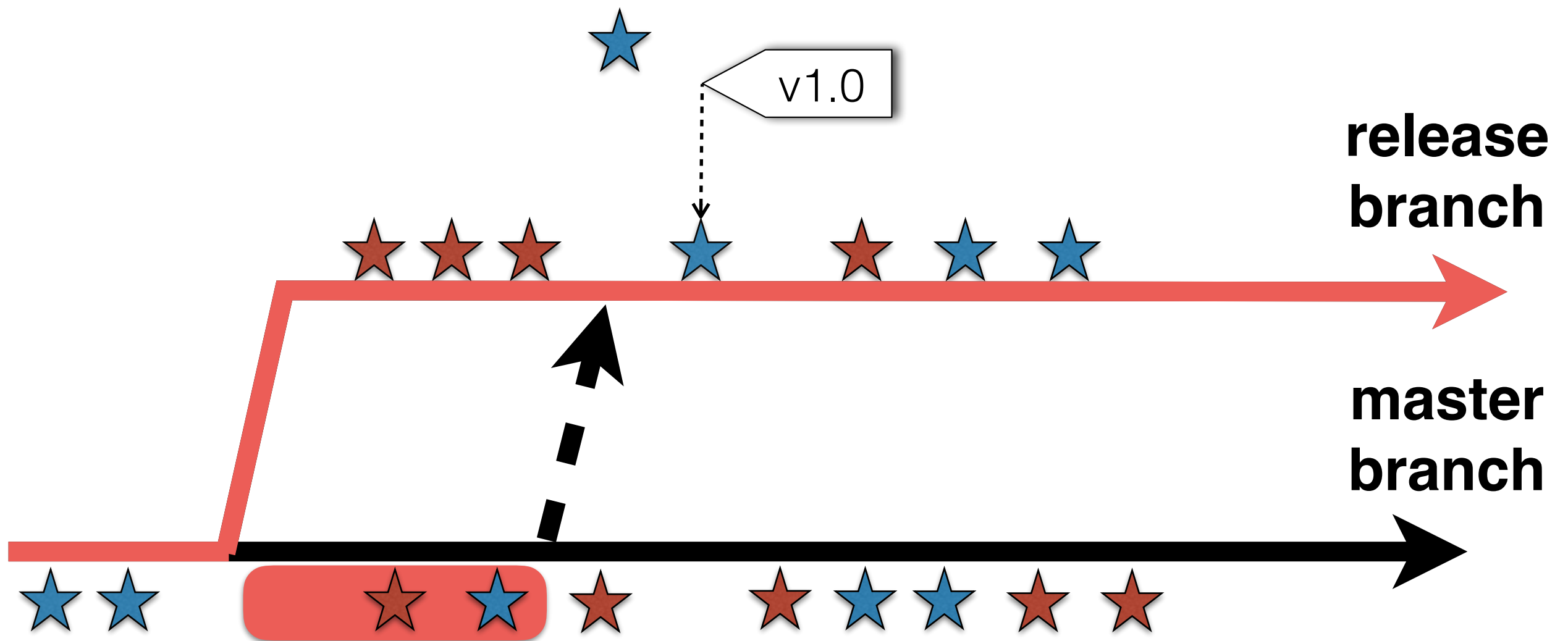
Solution: Select 1 Branch for Analysis and Expand Merge Commits



Commit types:

★ Feature ★ Fix ★ Merge

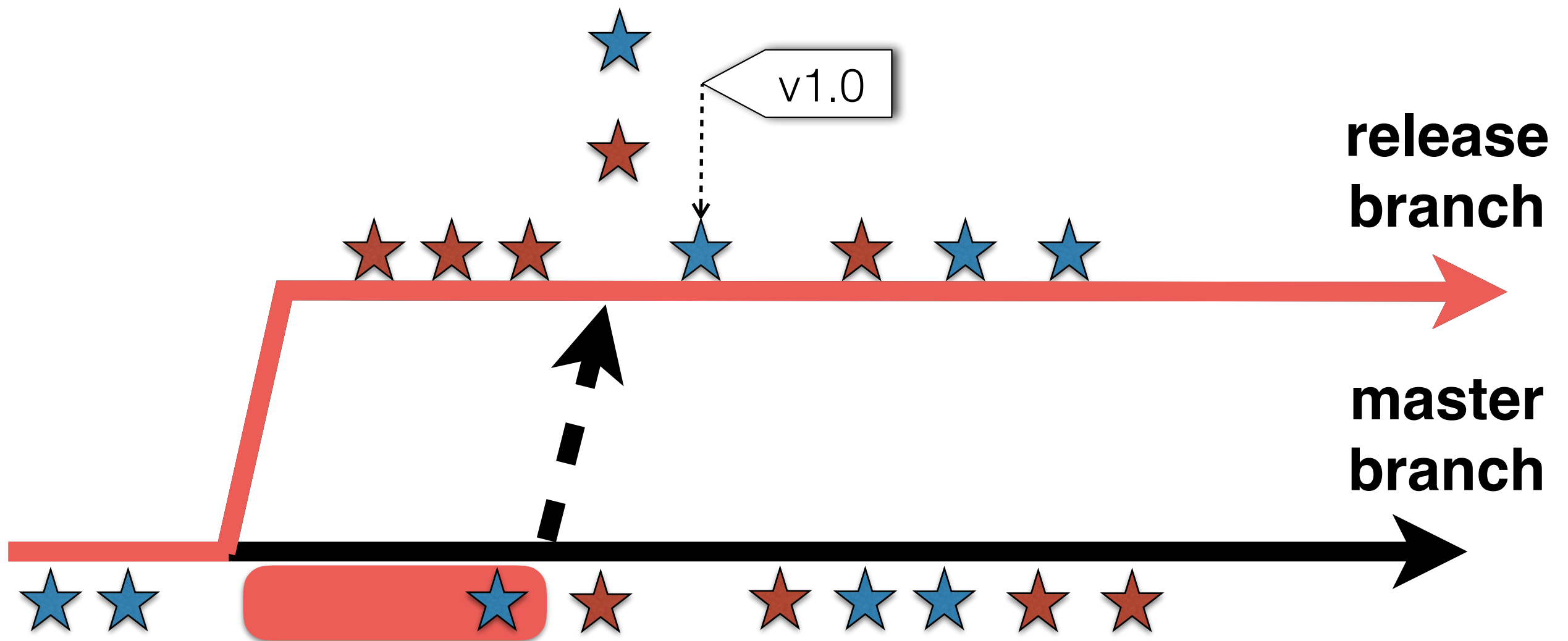
Solution: Select 1 Branch for Analysis and Expand Merge Commits



Commit types:

★ Feature ★ Fix ★ Merge

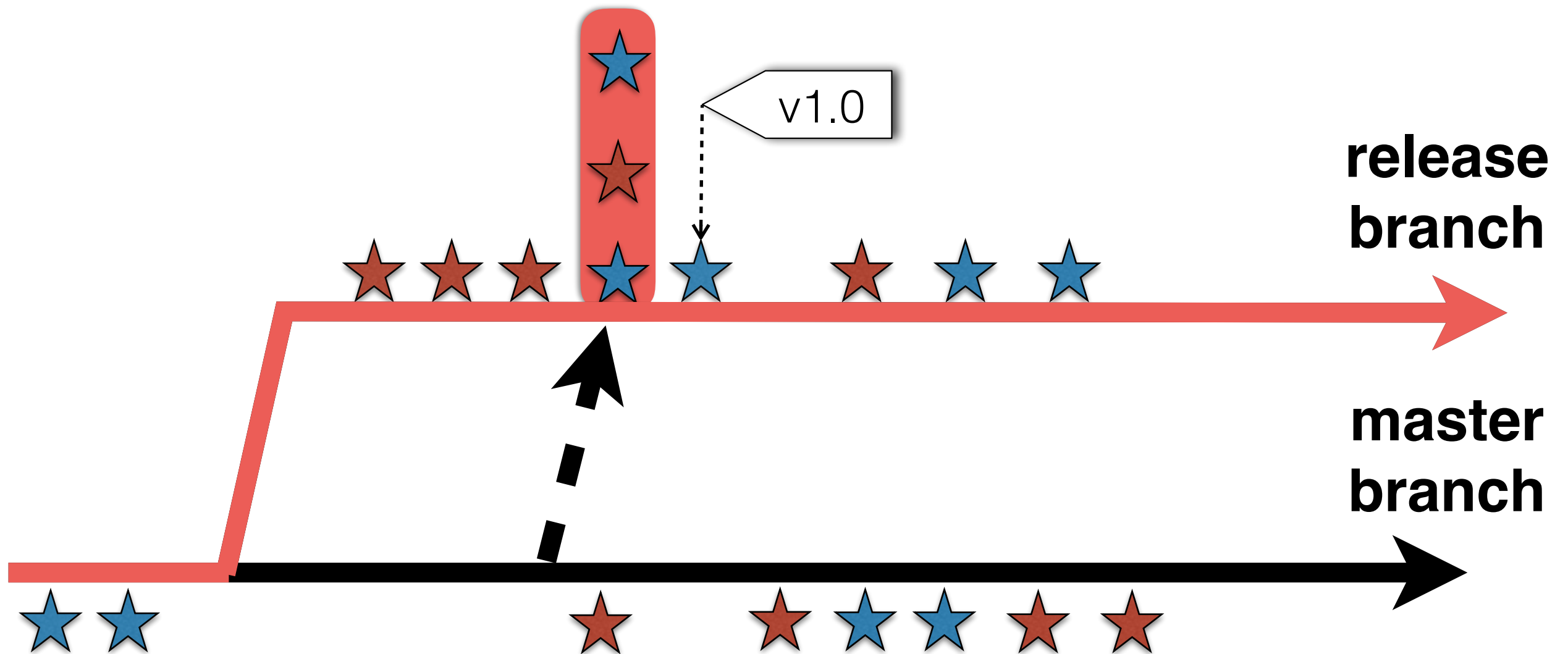
Solution: Select 1 Branch for Analysis and Expand Merge Commits



Commit types:

★ Feature ★ Fix ★ Merge

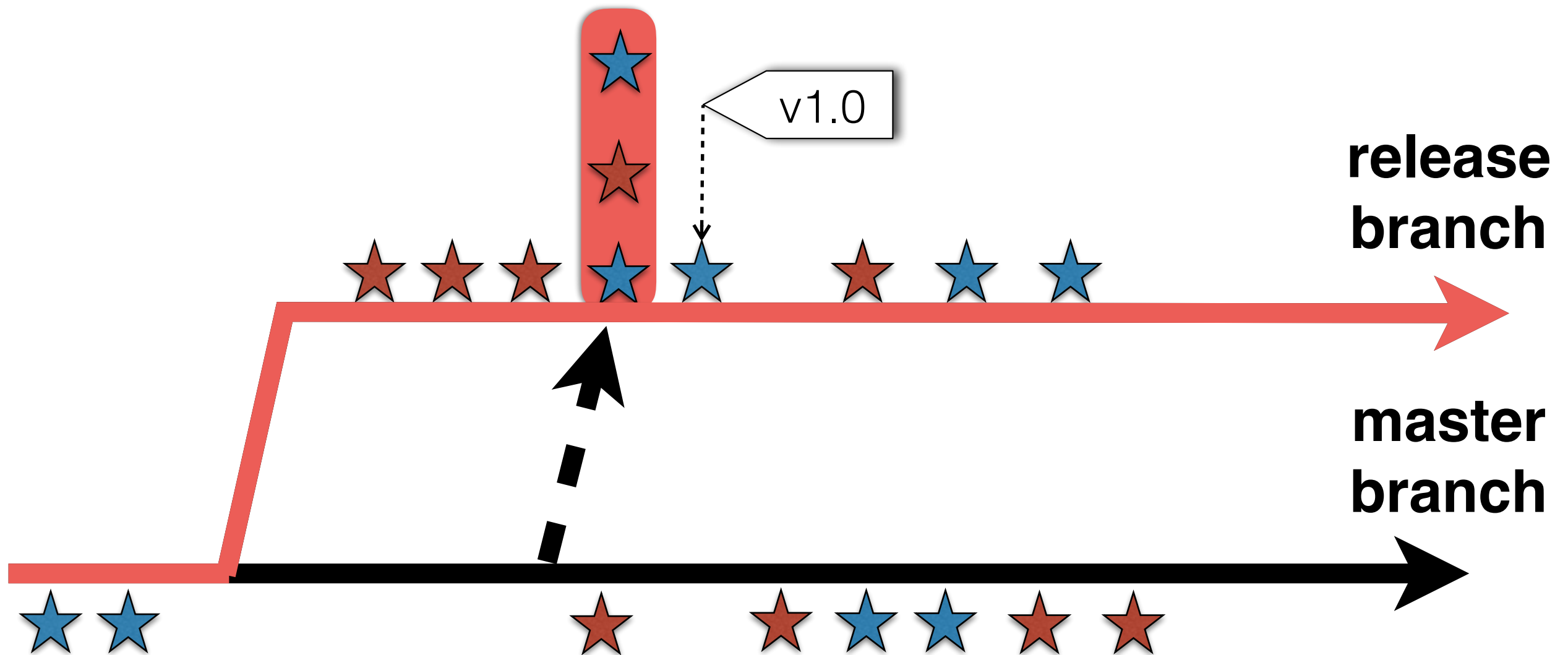
Solution: Select 1 Branch for Analysis and Expand Merge Commits



Commit types:

★ Feature ★ Fix ★ Merge

Solution: Select 1 Branch for Analysis and Expand Merge Commits

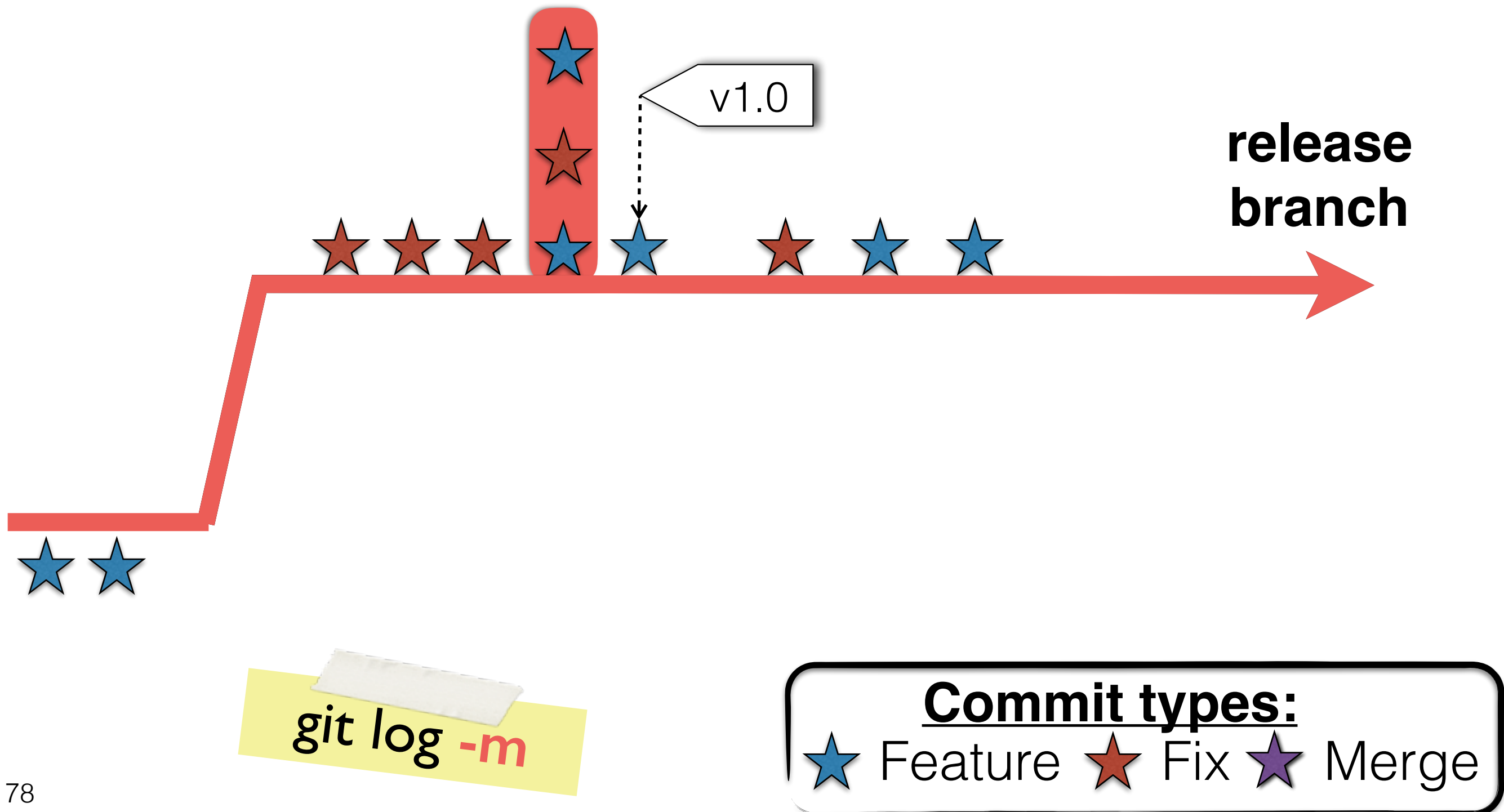


`git log -m`

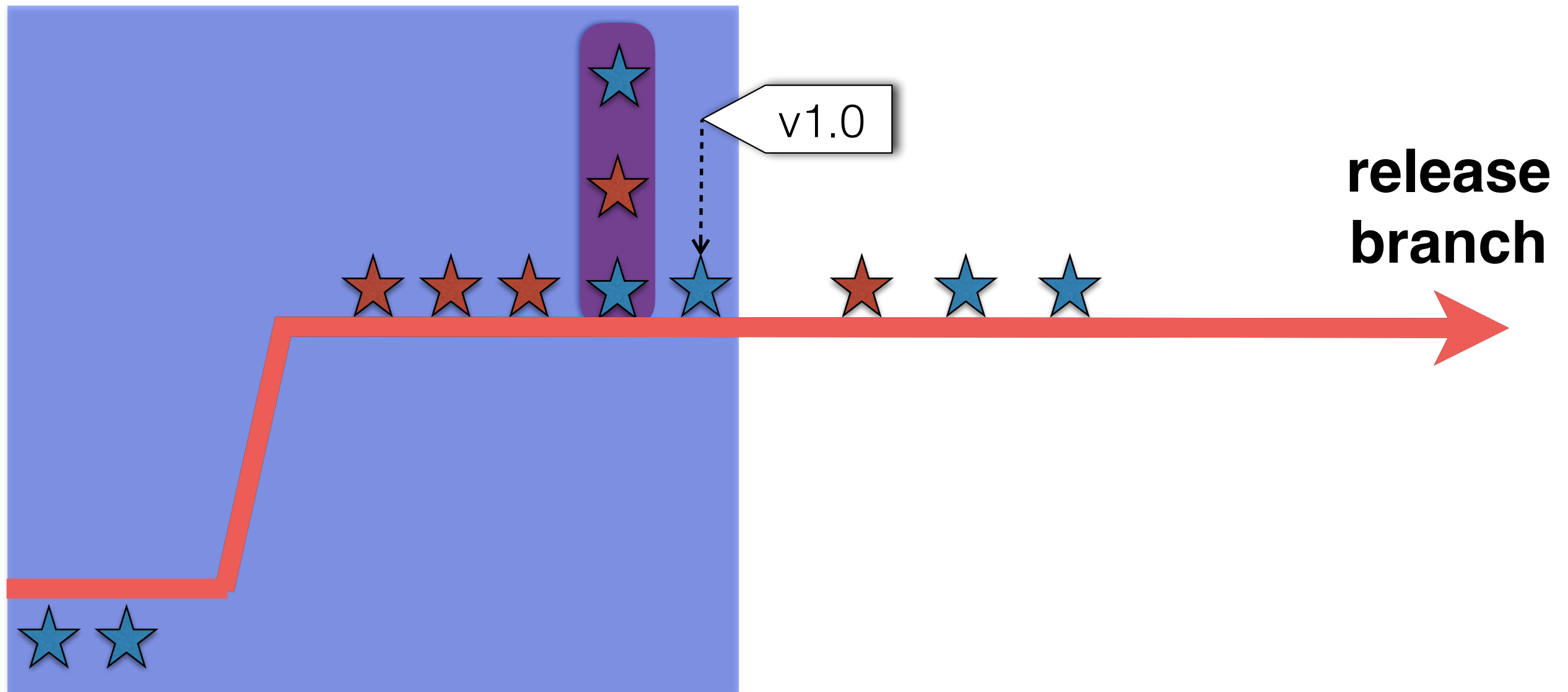
Commit types:

★ Feature ★ Fix ★ Merge

Solution: Select 1 Branch for Analysis and Expand Merge Commits



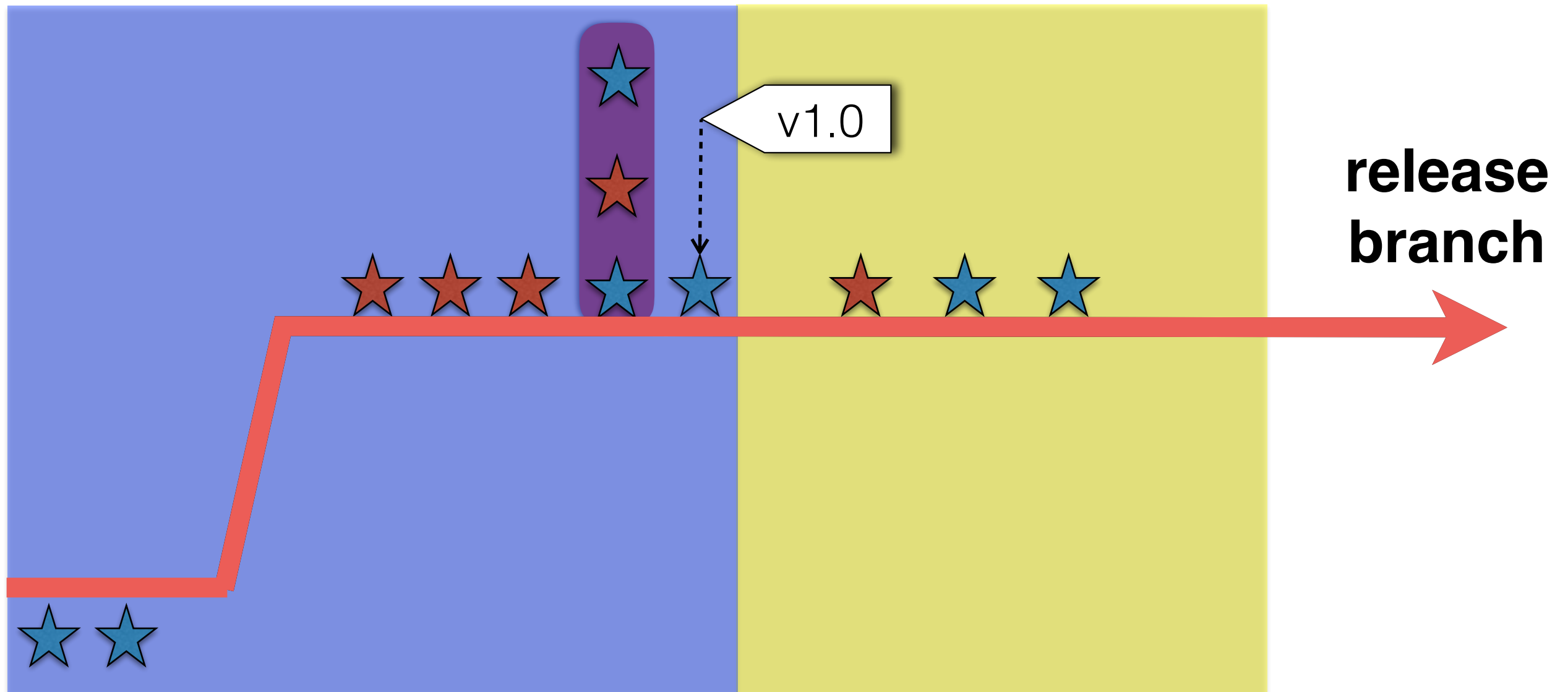
Solution: Select 1 Branch for Analysis and Expand Merge Commits



`git log -m`

Commit types:
★ Feature ★ Fix ★ Merge

Solution: Select 1 Branch for Analysis and Expand Merge Commits



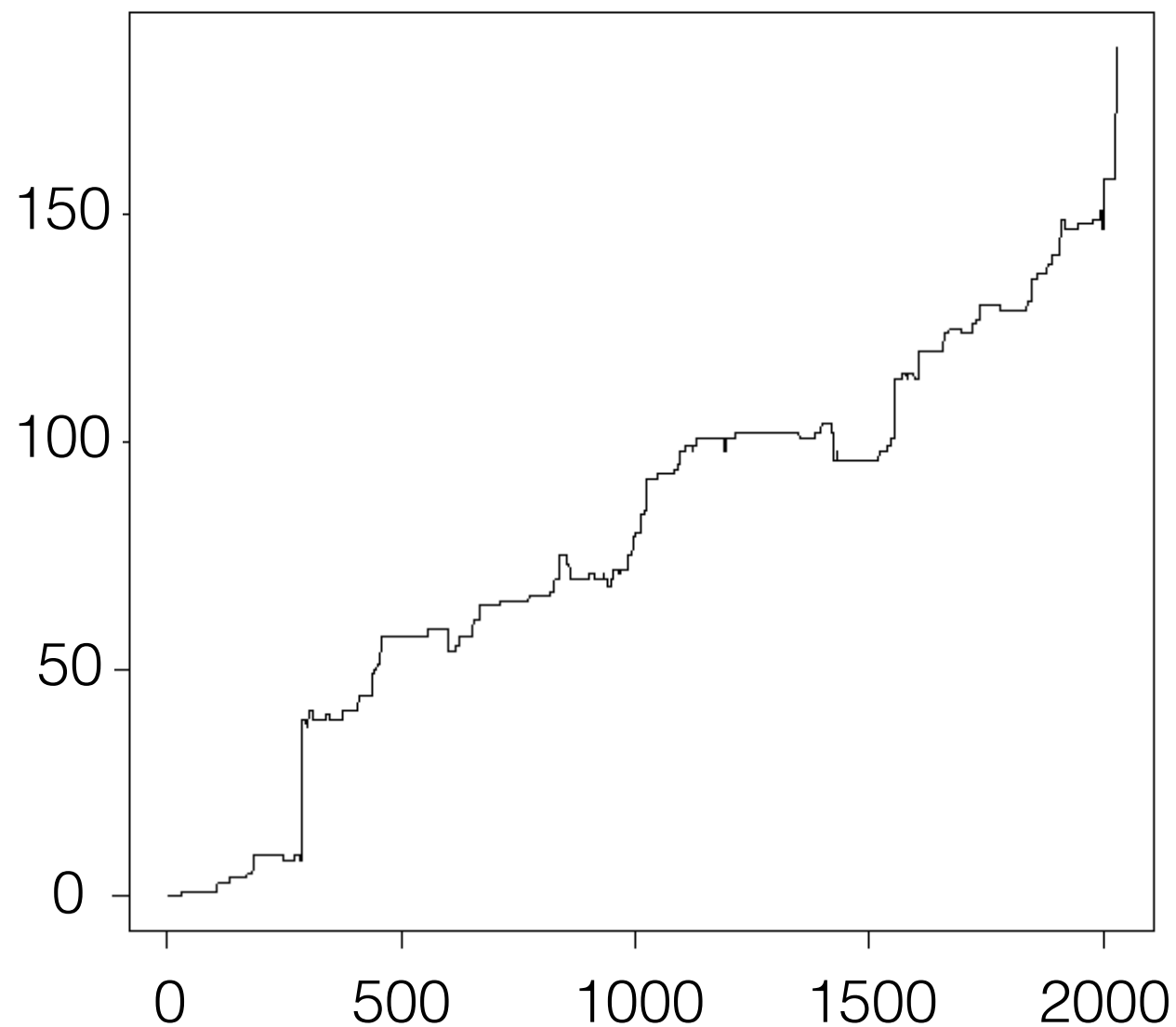
`git log -m`

Commit types:

★ Feature ★ Fix ★ Merge

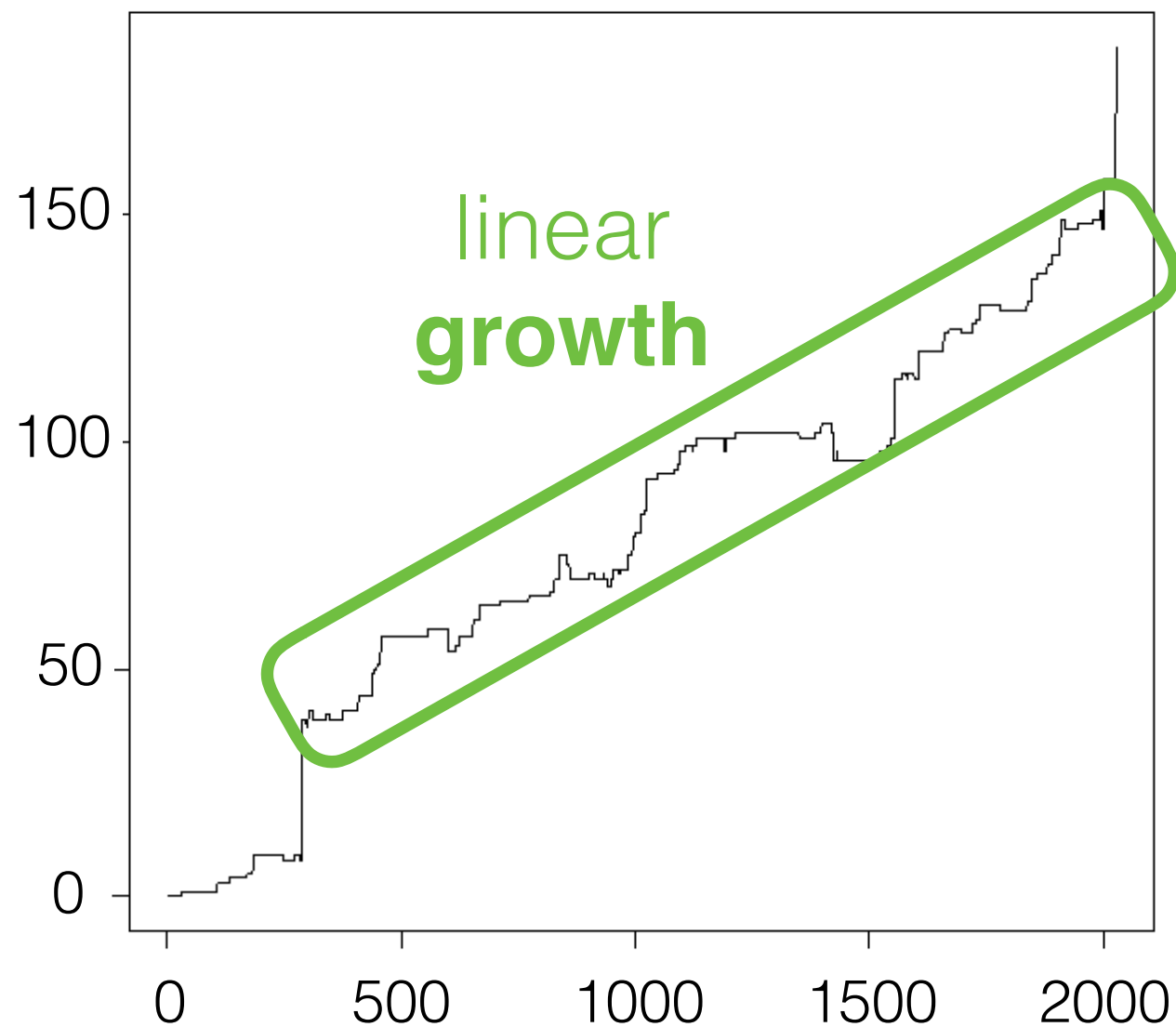
Evolution of #Files over Time

Evolution of #Files over Time



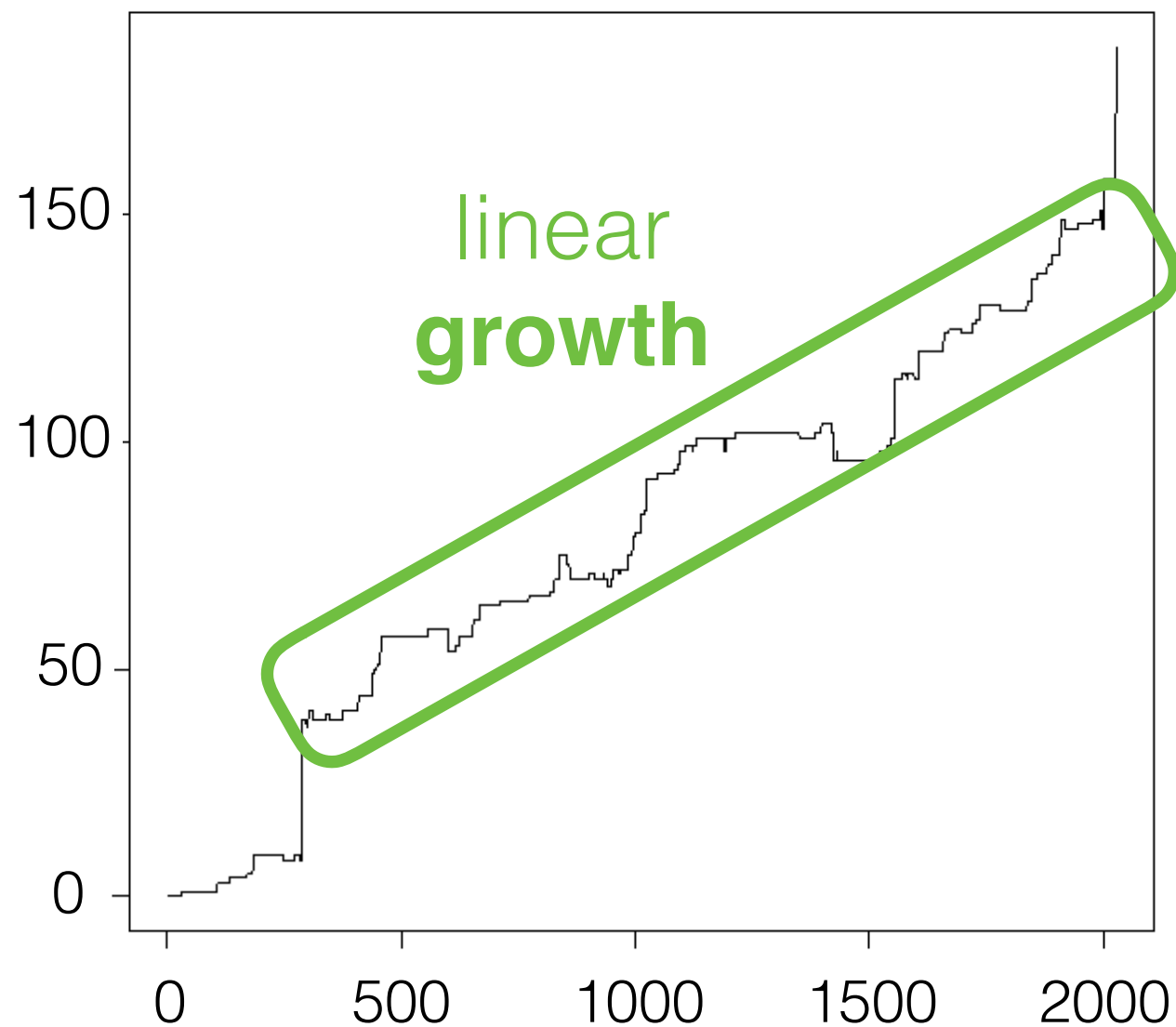
all commits, ordered
by author time (git log)

Evolution of #Files over Time

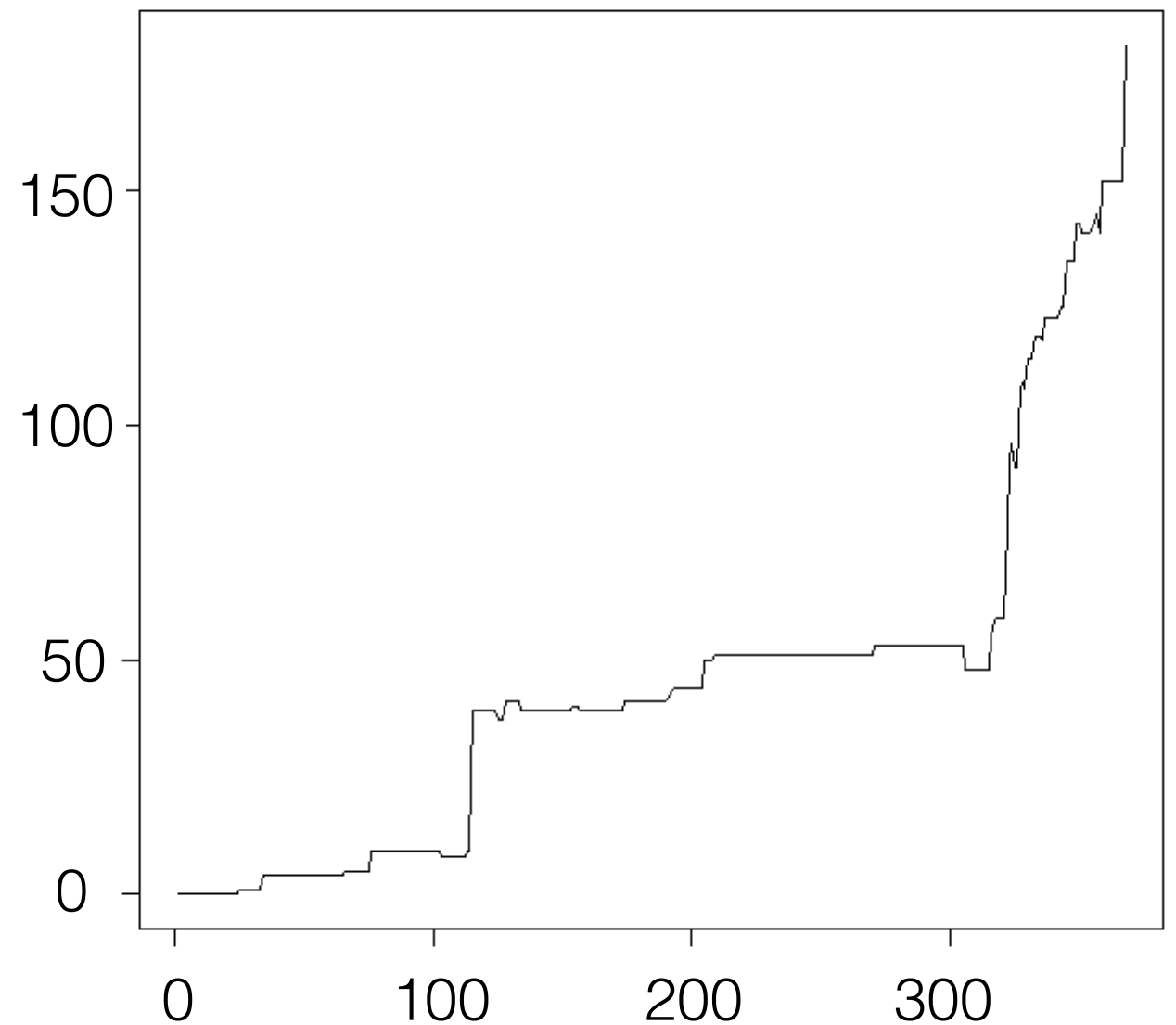


all commits, ordered
by author time (git log)

Evolution of #Files over Time

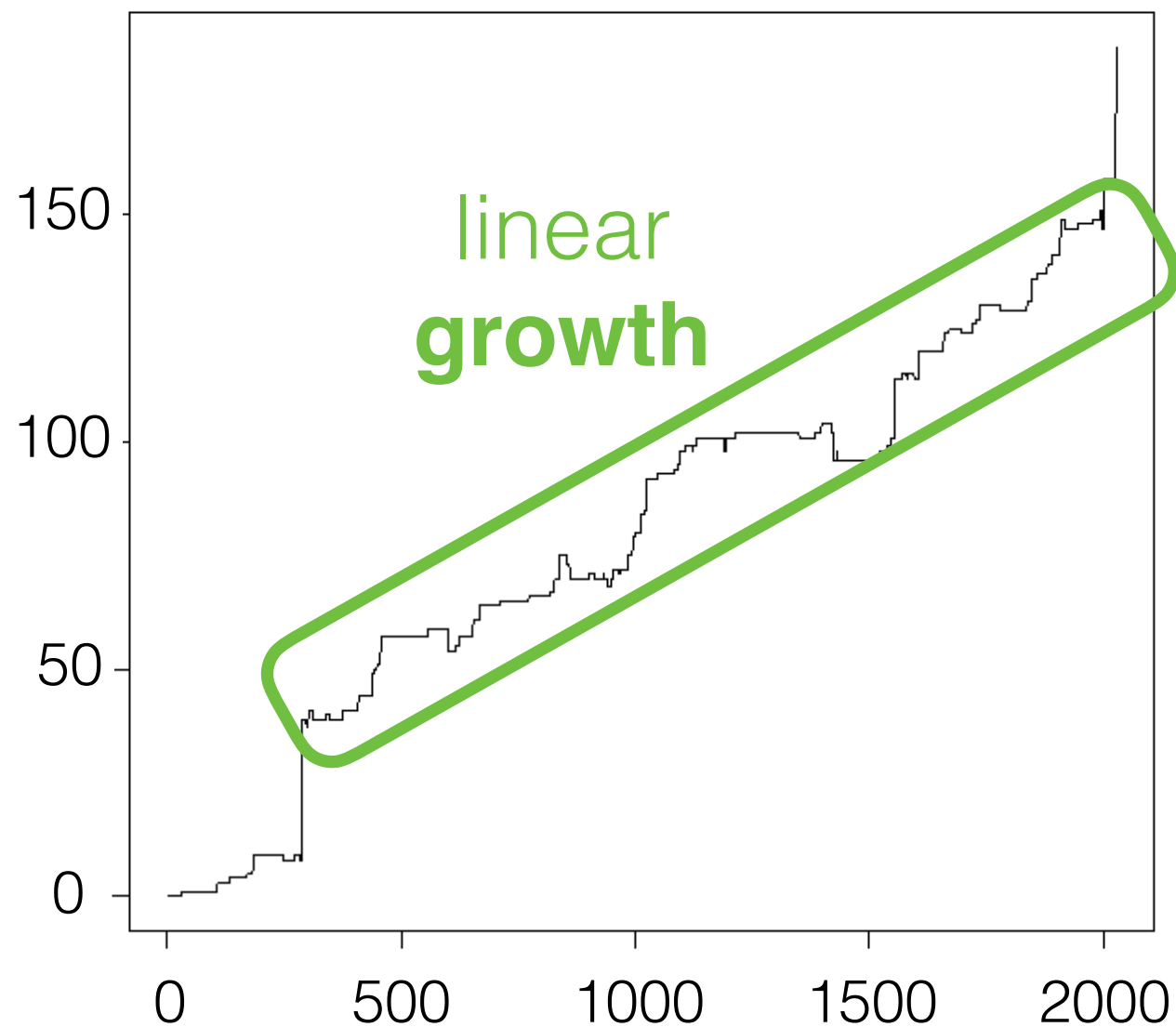


all commits, ordered
by author time (git log)

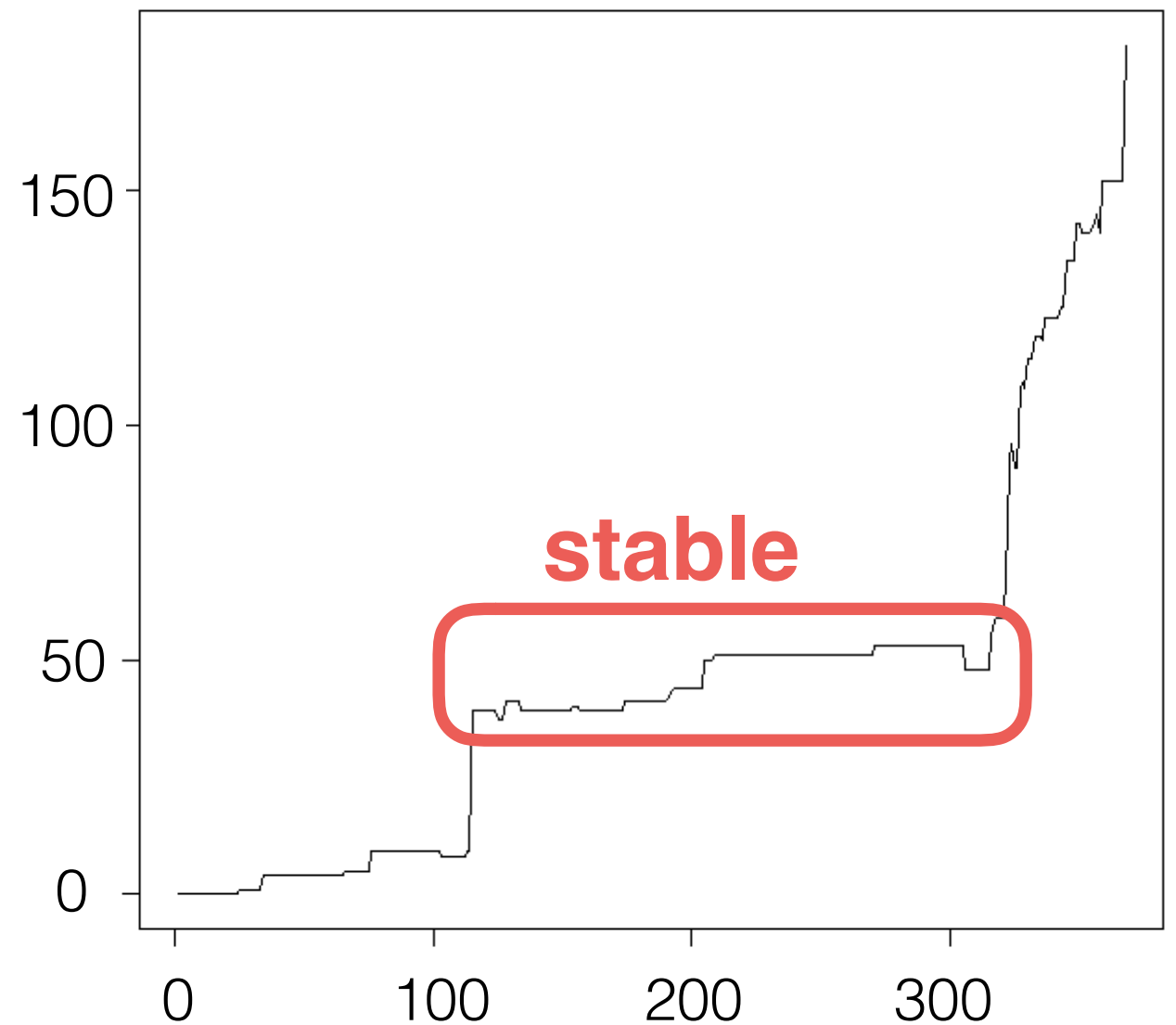


master branch, merge
commits expanded

Evolution of #Files over Time

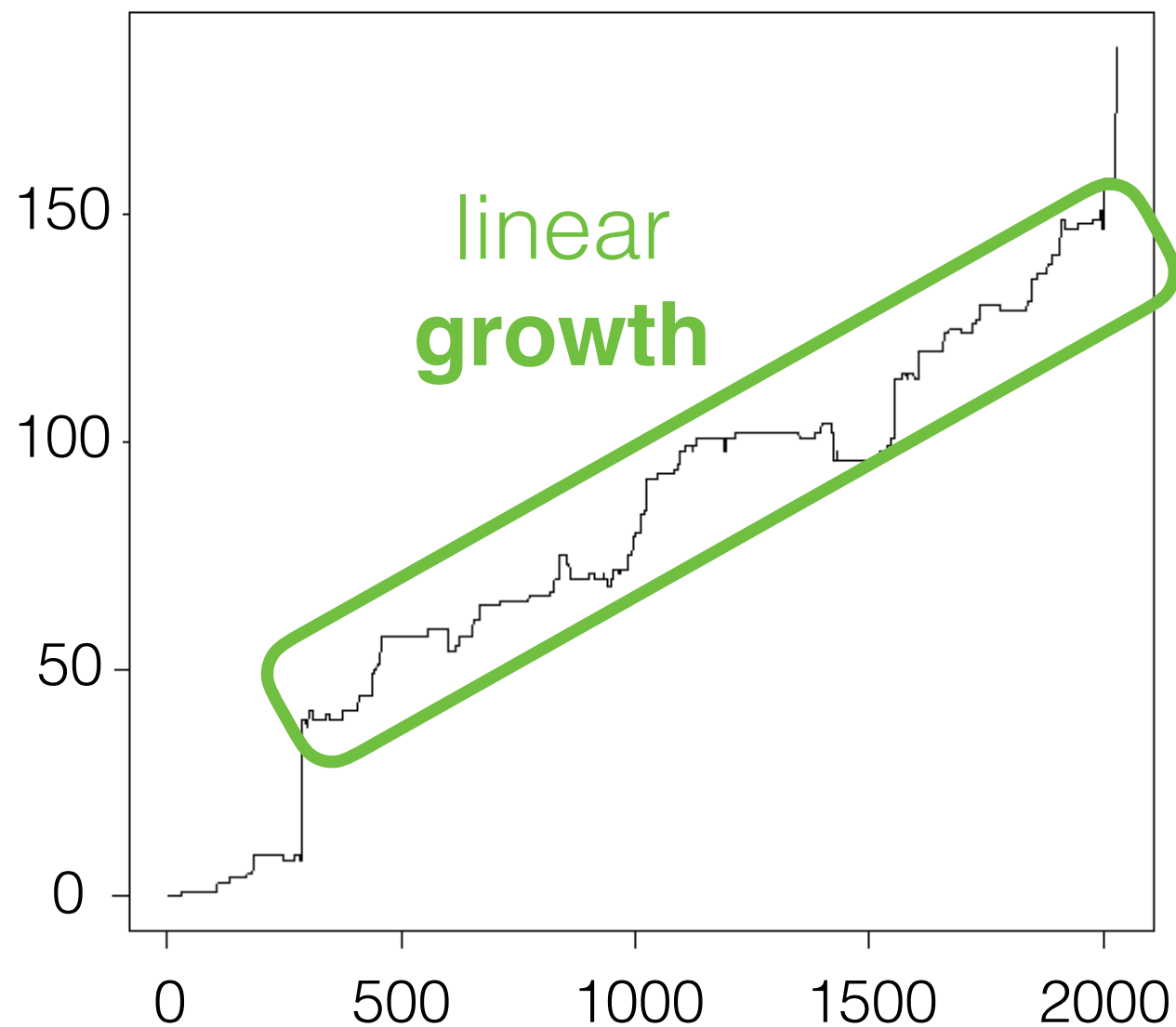


all commits, ordered
by author time (git log)

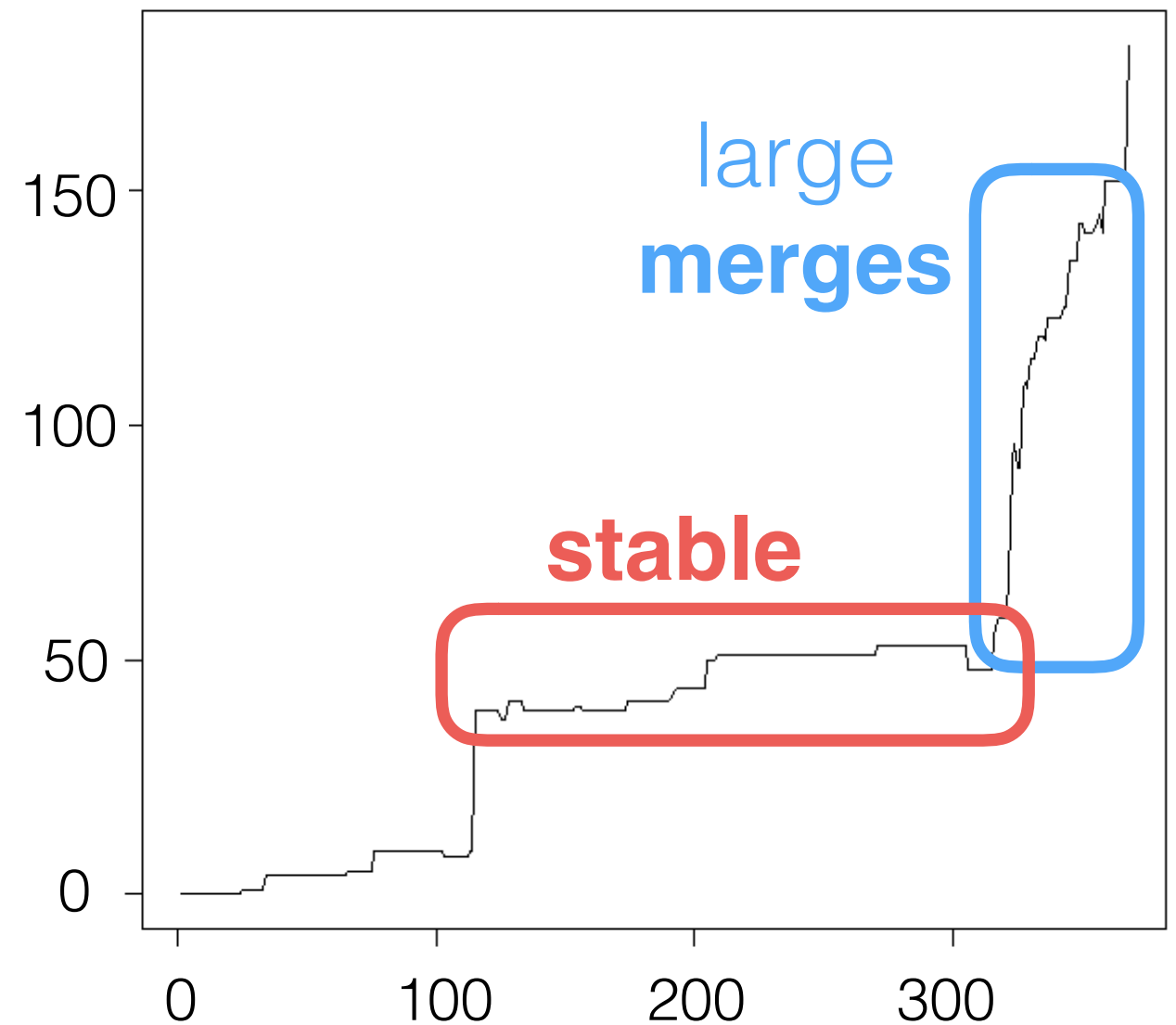


master branch, merge
commits expanded

Evolution of #Files over Time

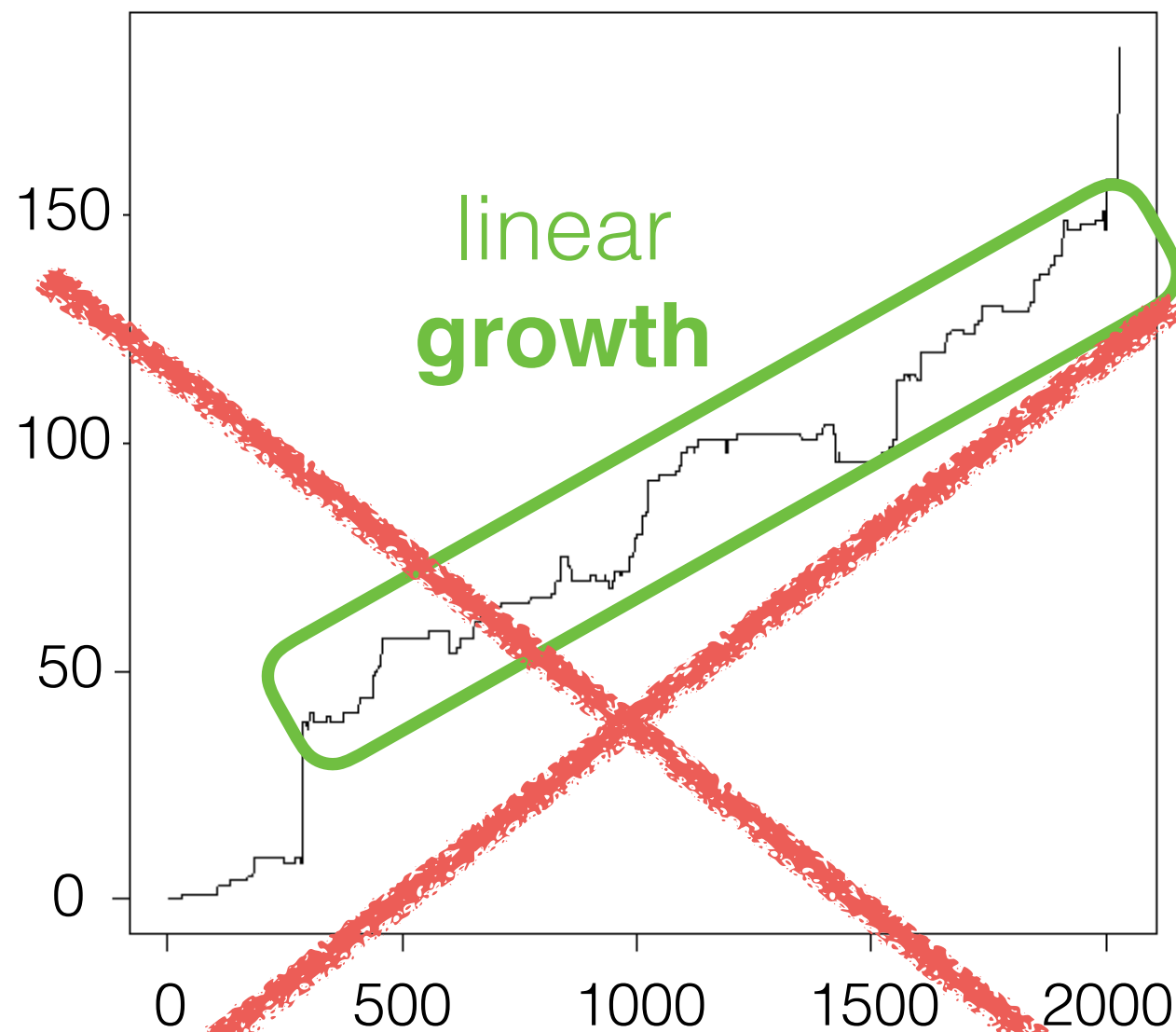


all commits, ordered
by author time (git log)

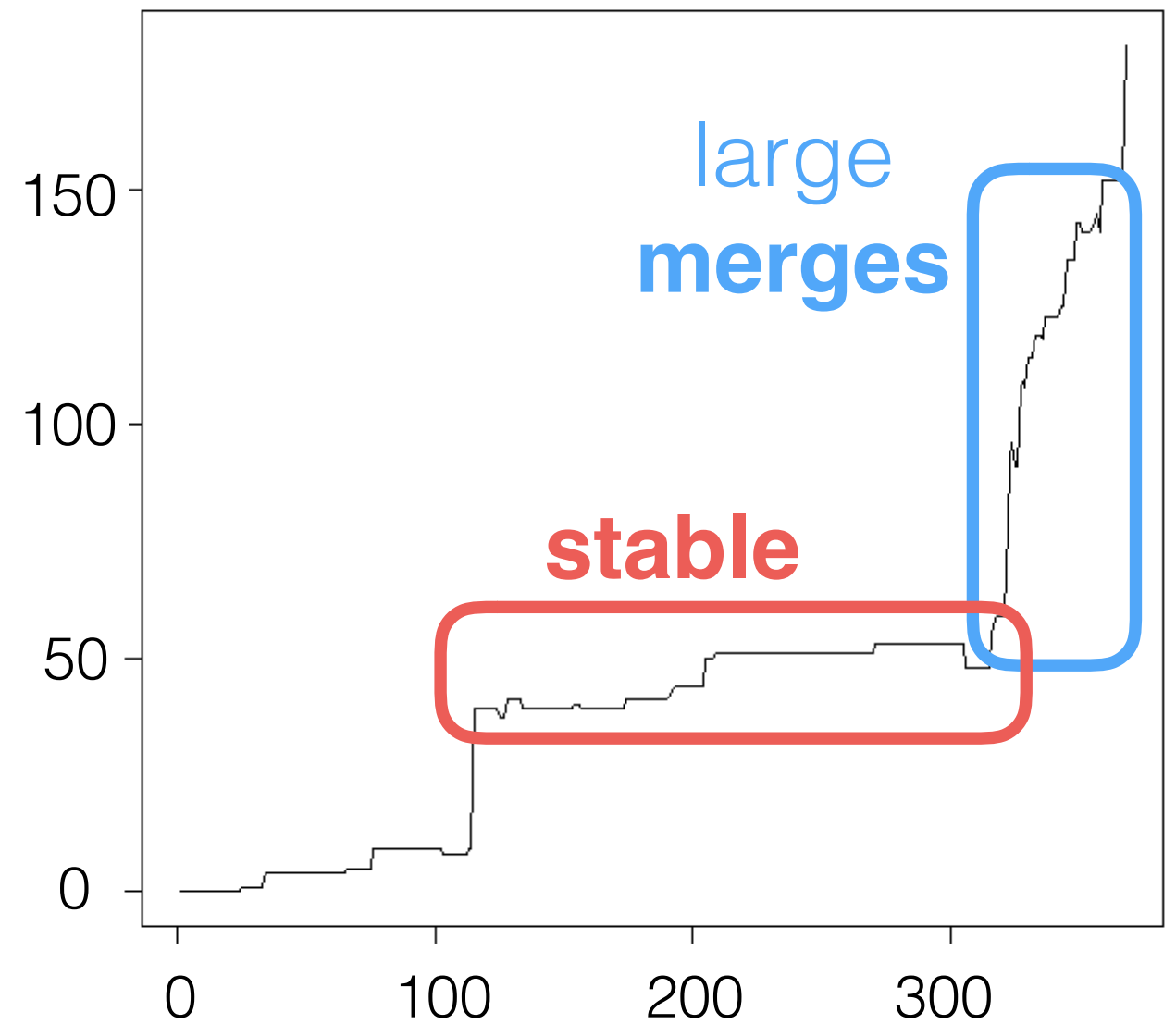


master branch, merge
commits expanded

Evolution of #Files over Time



~~all commits, ordered
by author time (git log)~~



master branch, merge
commits expanded

3. All Files are Equal



~~3. All Files are Equal~~





This popular web browser
has an enormous codebase!
>30M lines!



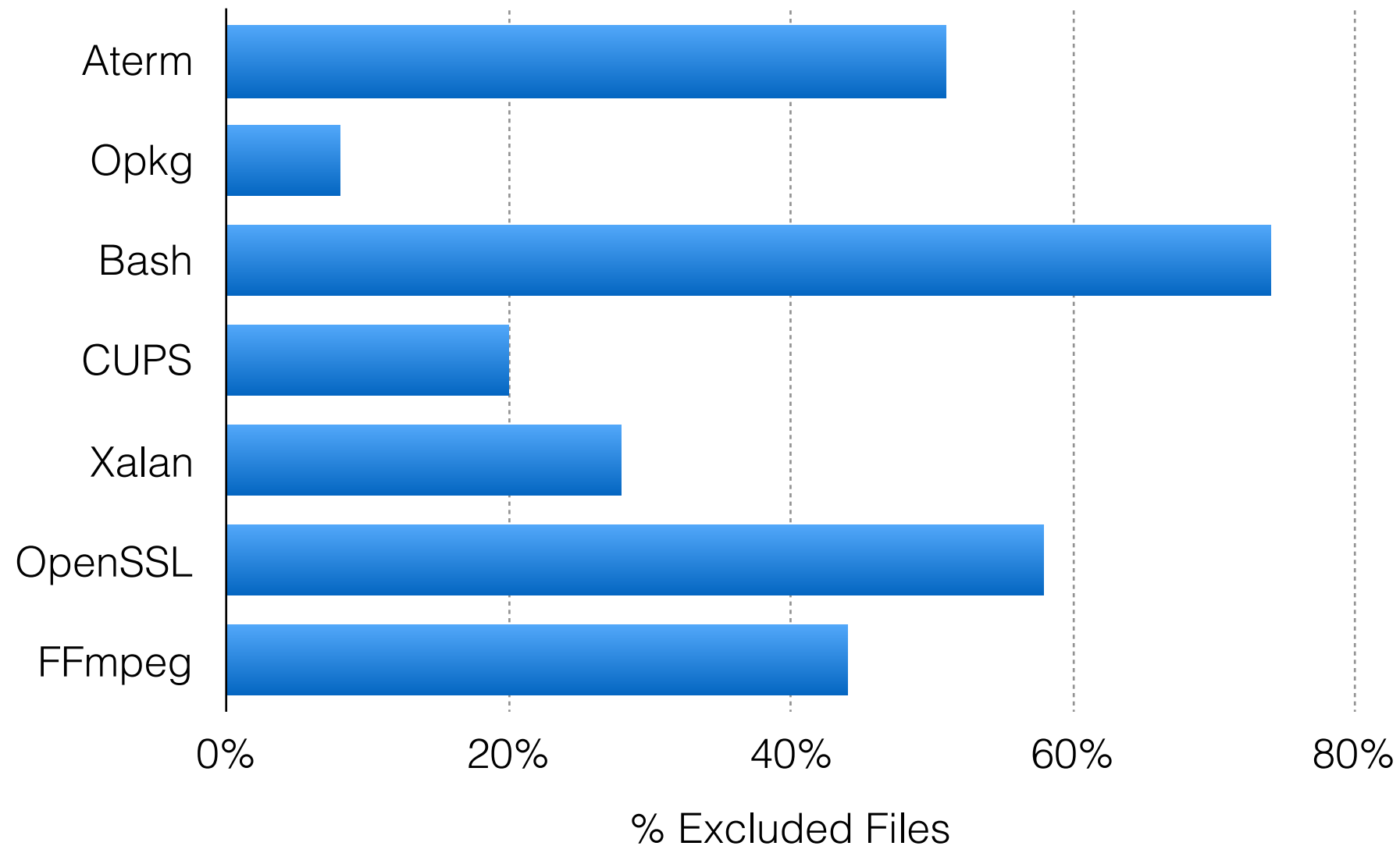


This popular web browser
has an enormous codebase!
>30M lines!



Yes, but the codebase
contains **several projects!**
The **build configuration**
decides which one is
produced!

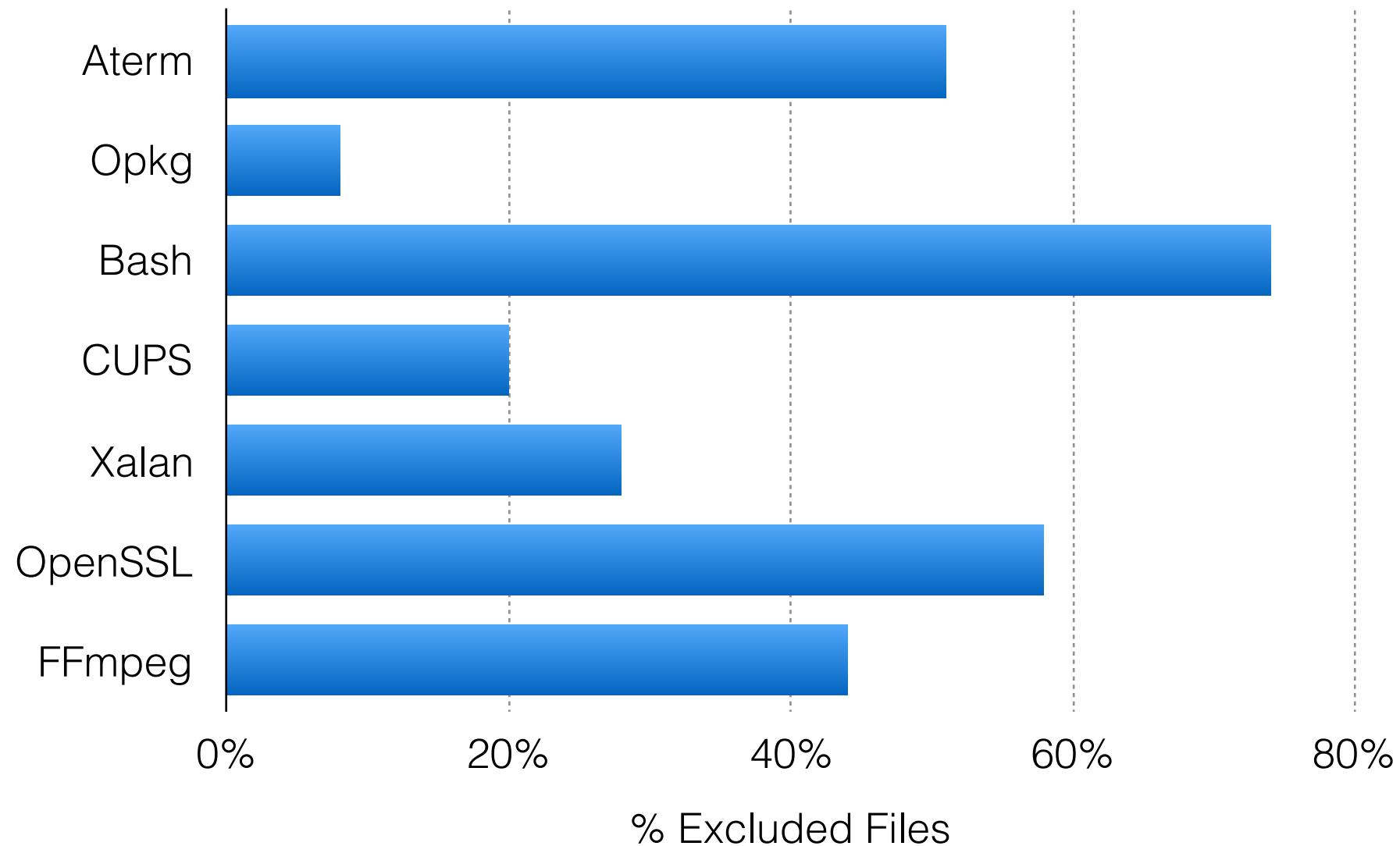
Many files are conditionally included in deliverables



Tracing Software Build Processes to Uncover License Compliance Inconsistencies

S. van der Berg *et al.*
[ASE 2014]

Many files are conditionally included in deliverables



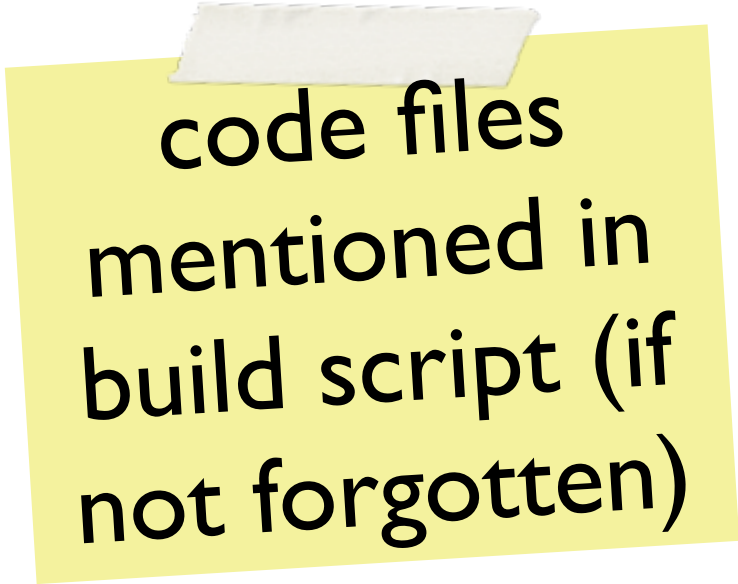
Tracing Software Build Processes to Uncover License Compliance Inconsistencies

S. van der Berg *et al.*
[ASE 2014]

... because of **feature** selection, **operating system/hardware** configuration, **dead** code, ...

Solution: Pick Configuration, Run the Build and Check Built Files

Solution: Pick Configuration, Run the Build and Check Built Files



code files
mentioned in
build script (if
not forgotten)

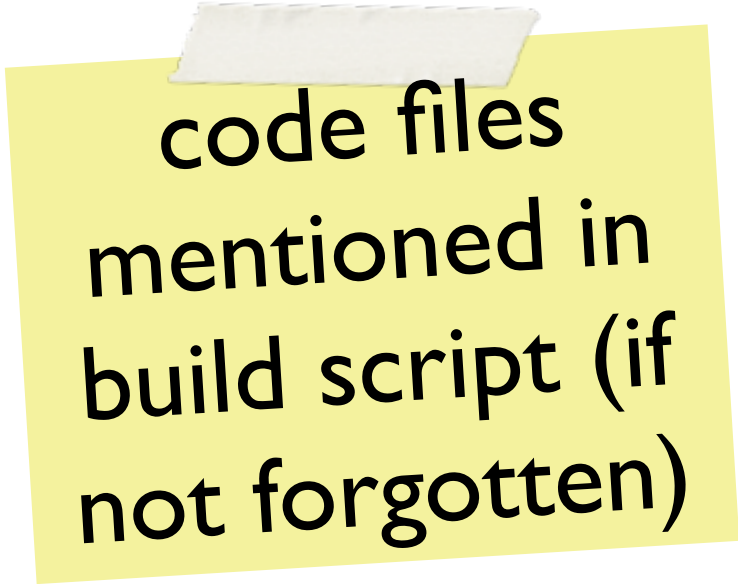
<http://mcis.polymtl.ca/makao.html>

**Design recovery and
maintenance of build systems**

B. Adams *et al.*

[ICSM 2007]

Solution: Pick Configuration, Run the Build and Check Built Files



code files
mentioned in
build script (if
not forgotten)

and/or

<http://mcis.polymtl.ca/makao.html>

**Design recovery and
maintenance of build systems**

B. Adams *et al.*

[ICSM 2007]

Solution: Pick Configuration, Run the Build and Check Built Files

code files
mentioned in
build script (if
not forgotten)

and/or

files accessed by
systems calls (strace;
can be noisy!)

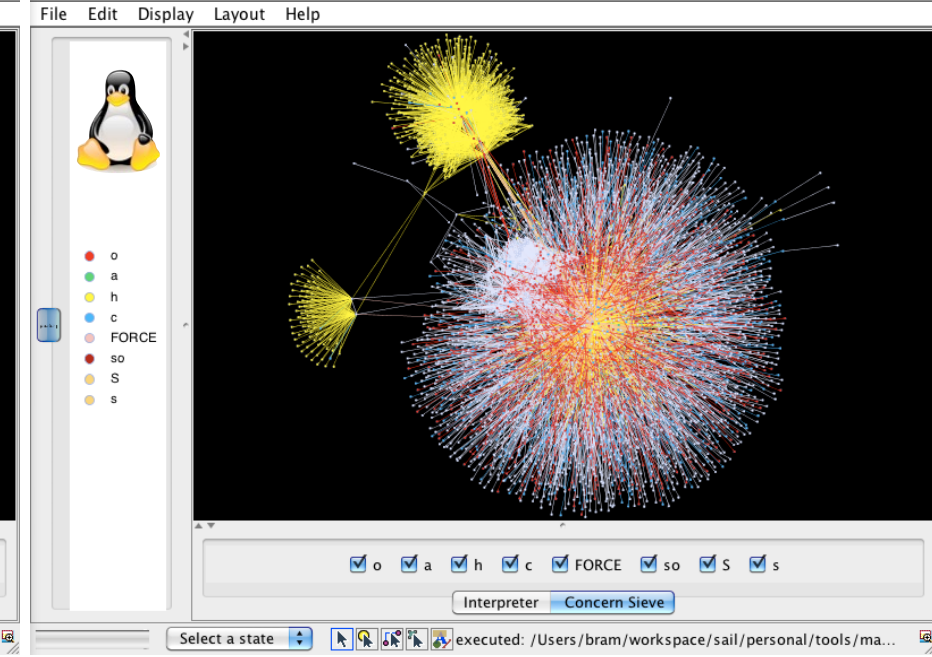
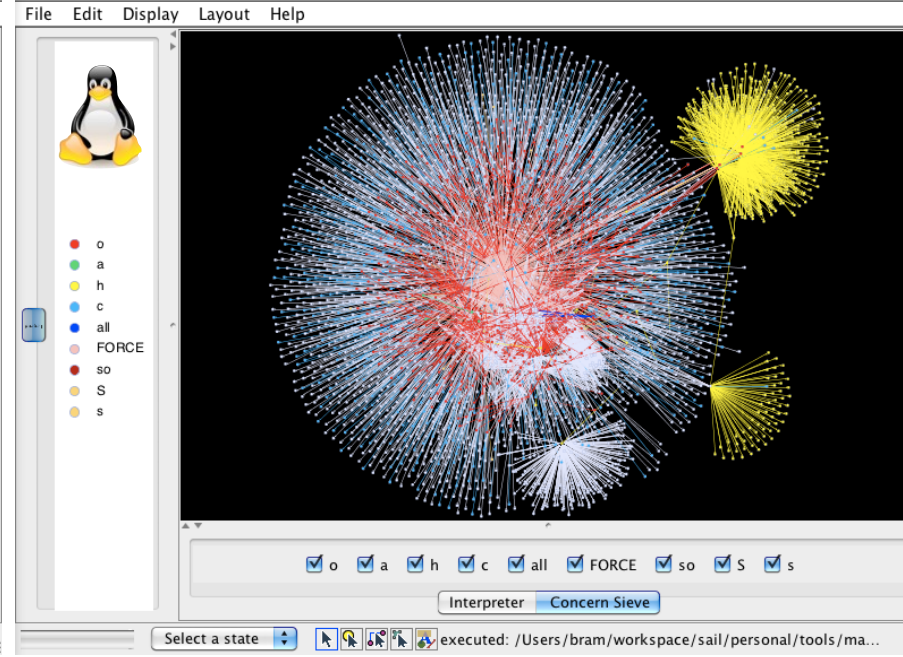
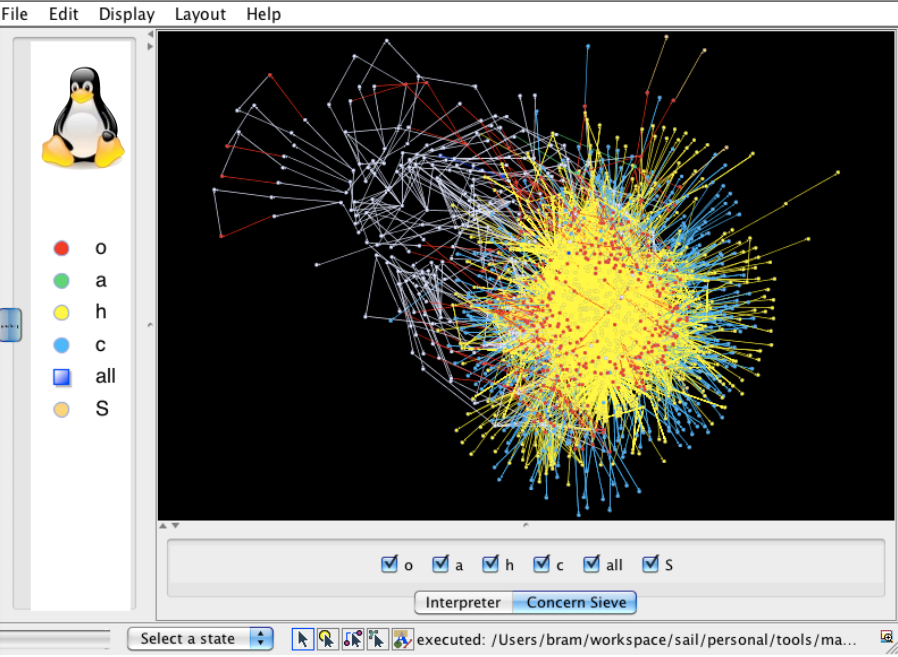
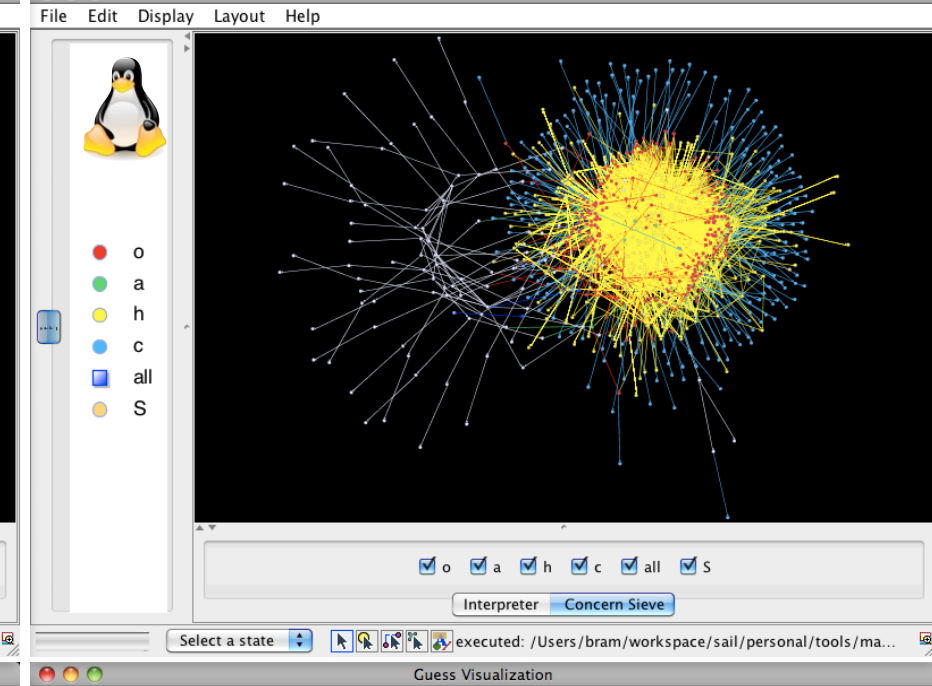
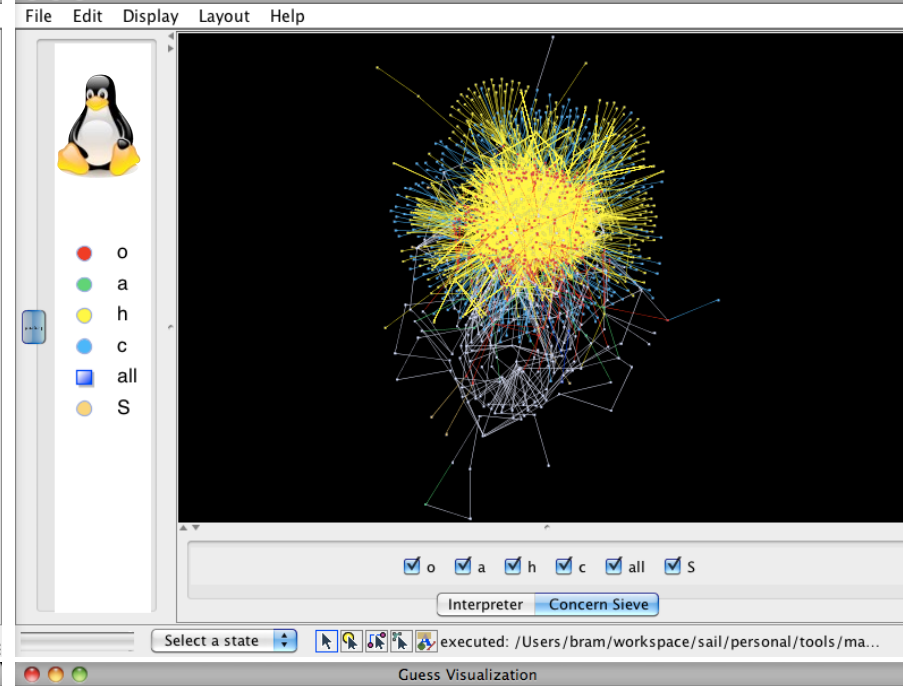
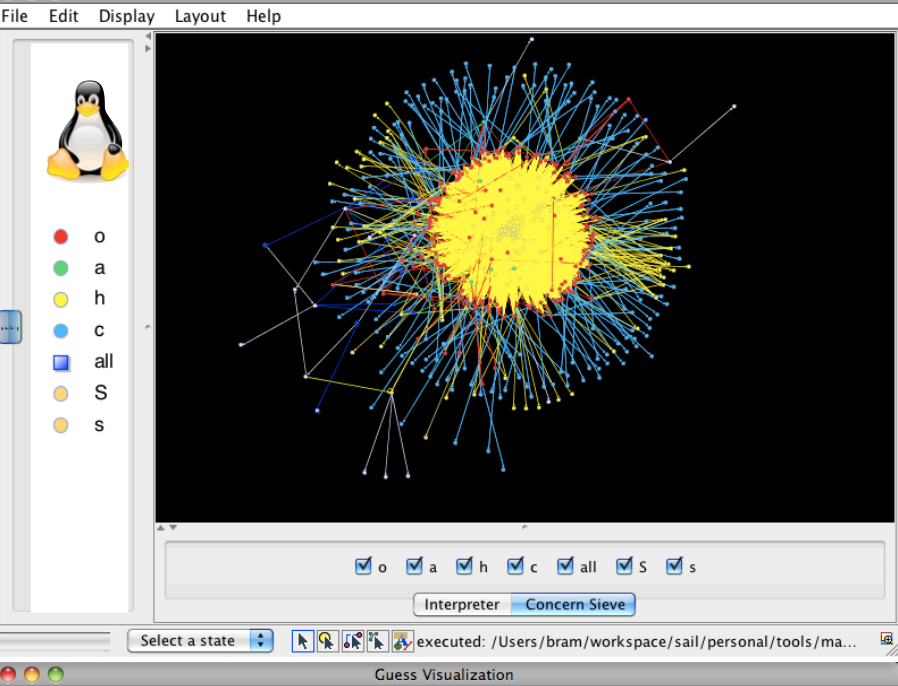
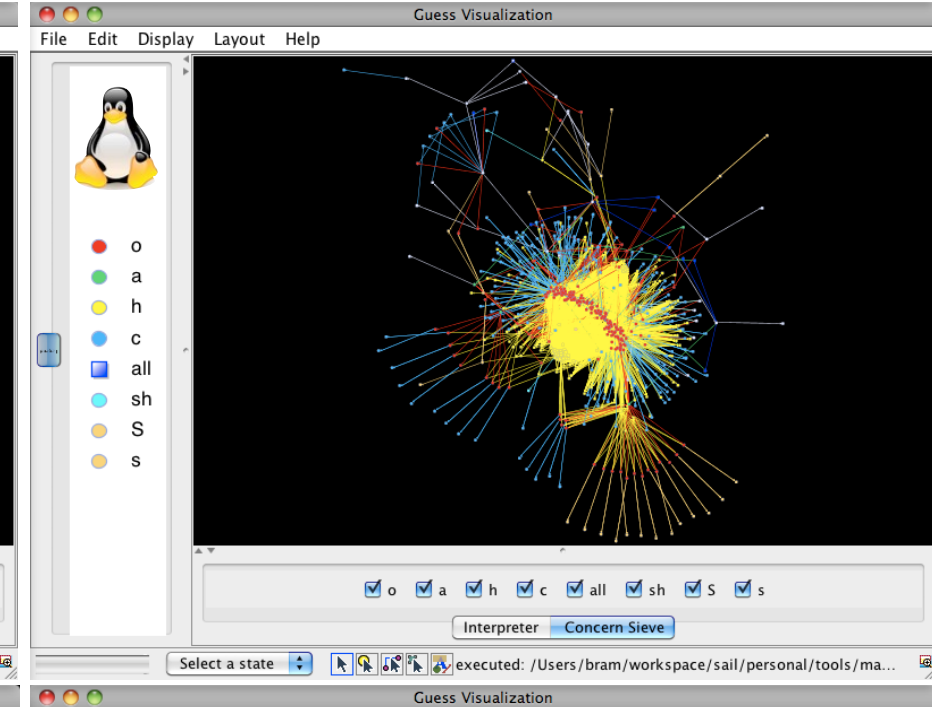
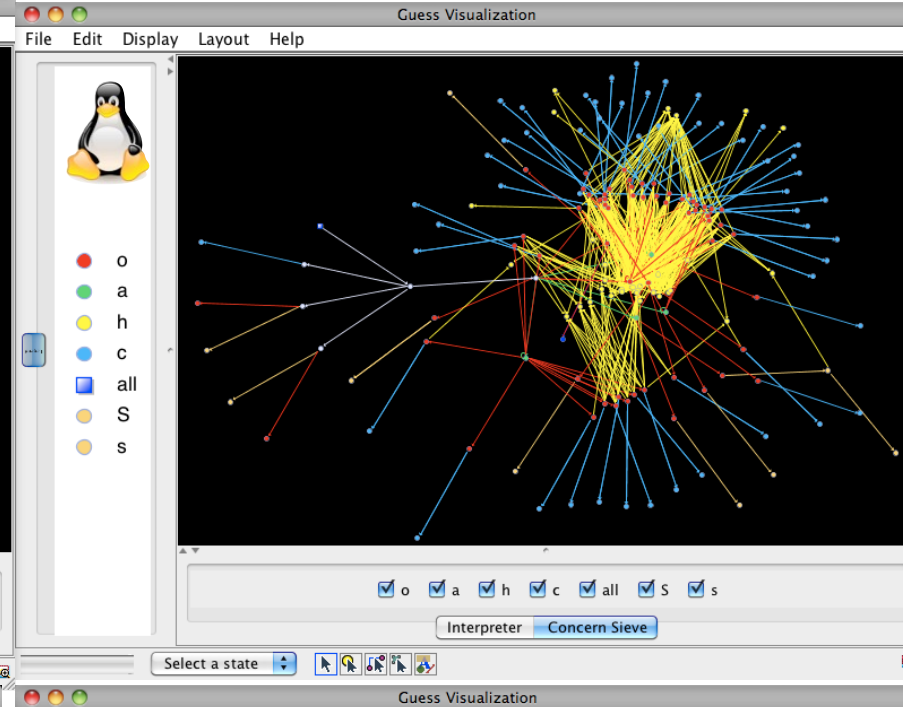
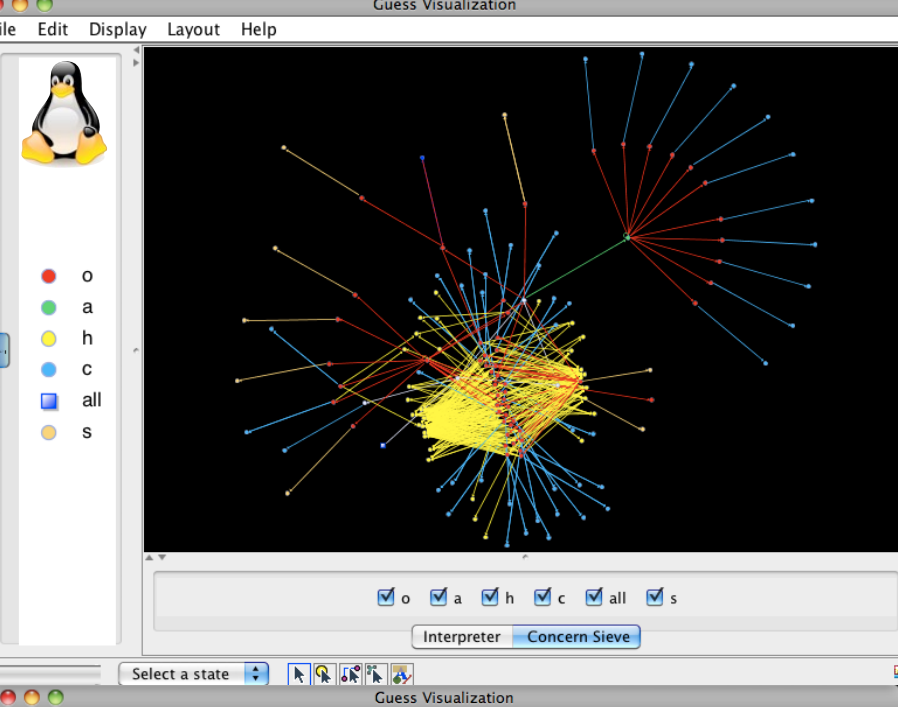
<http://mcis.polymtl.ca/makao.html> <https://github.com/smcintosh/bee>

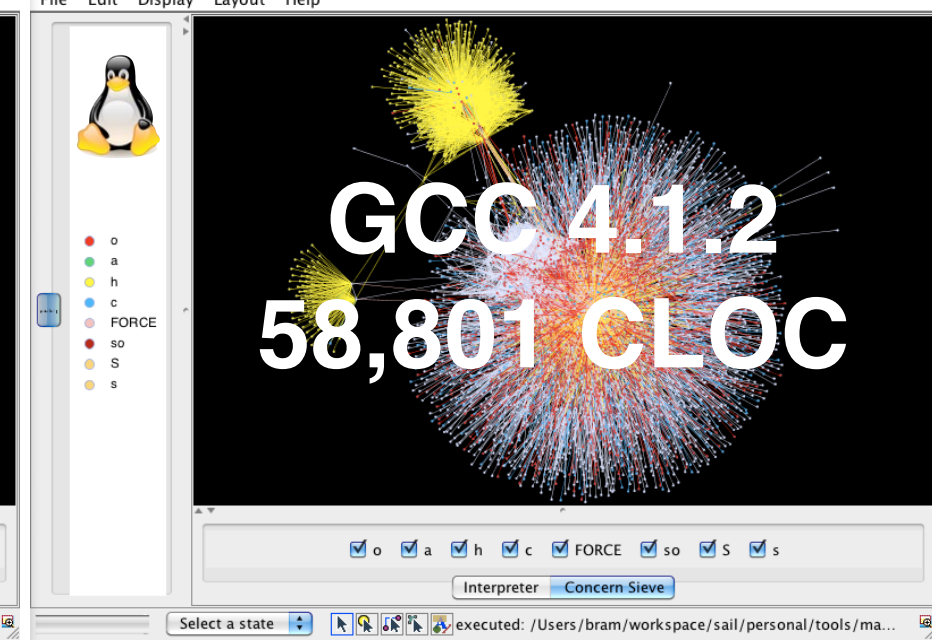
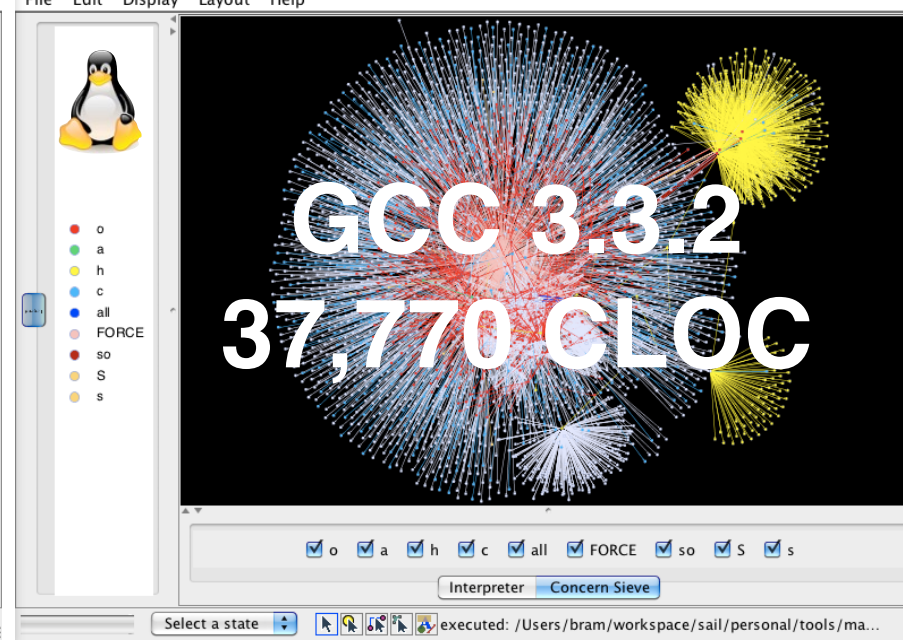
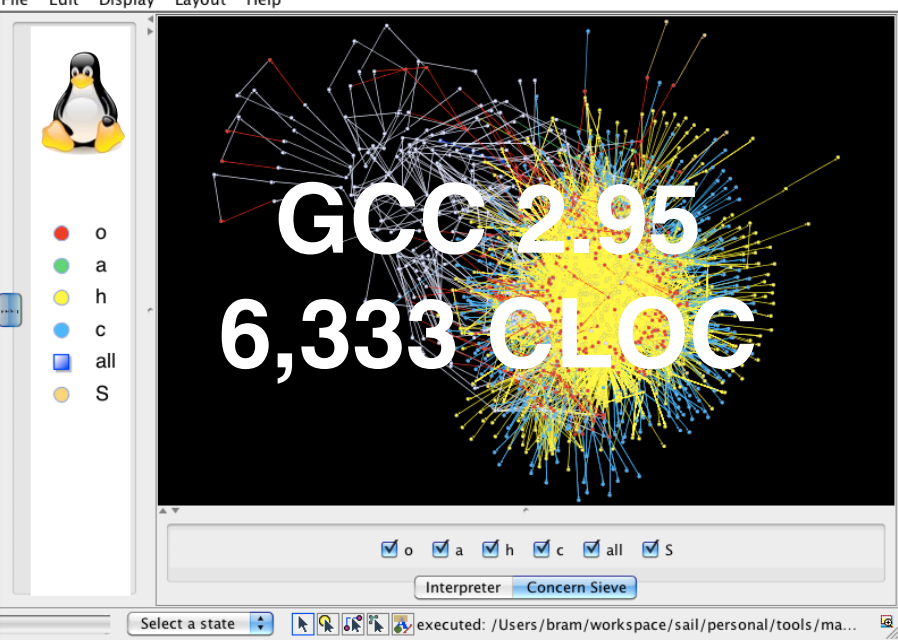
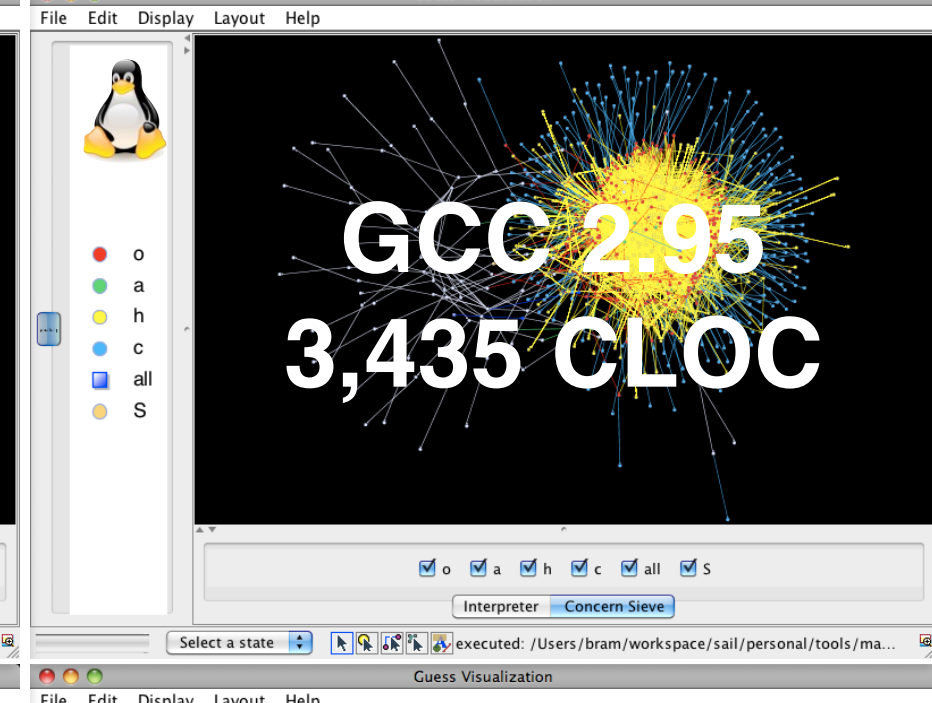
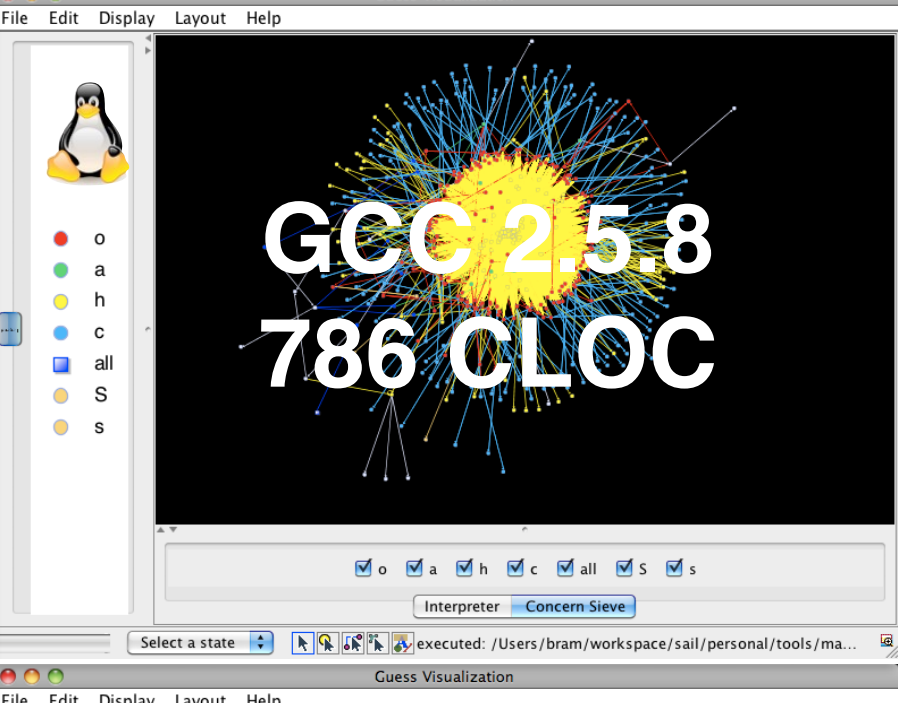
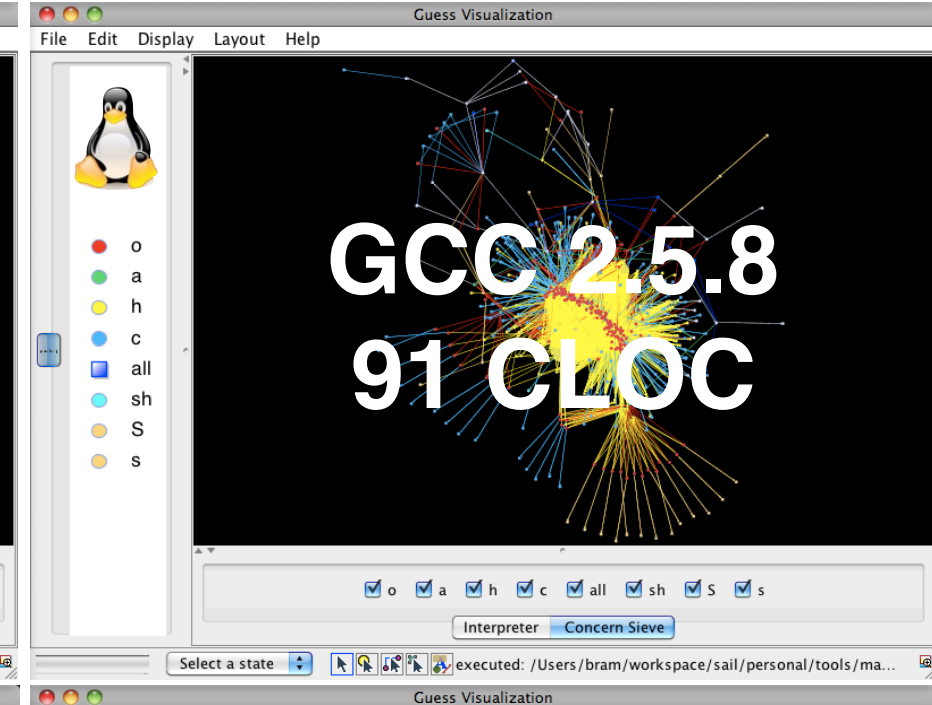
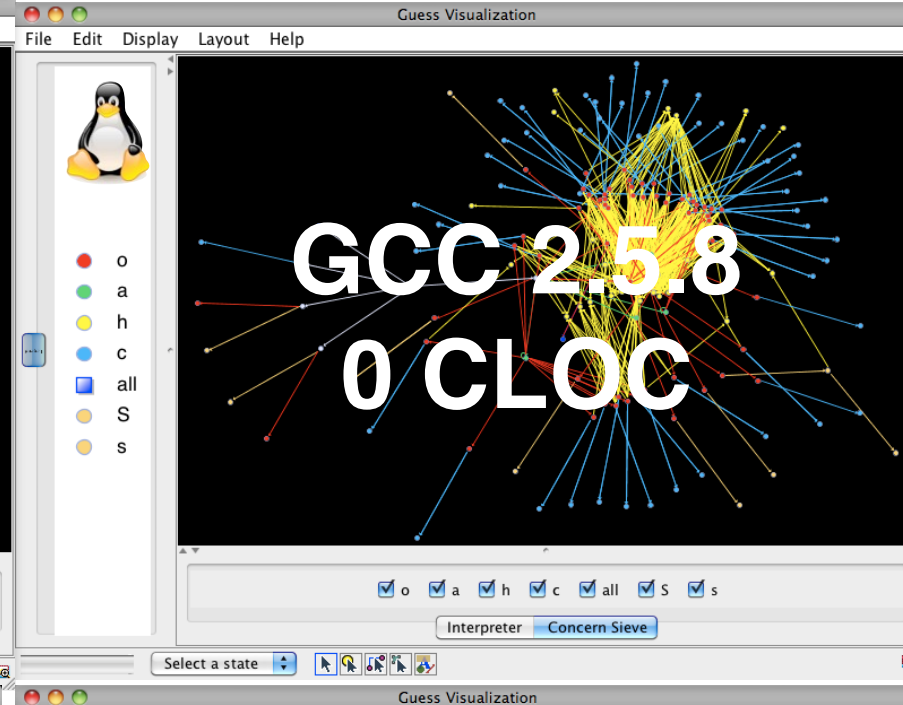
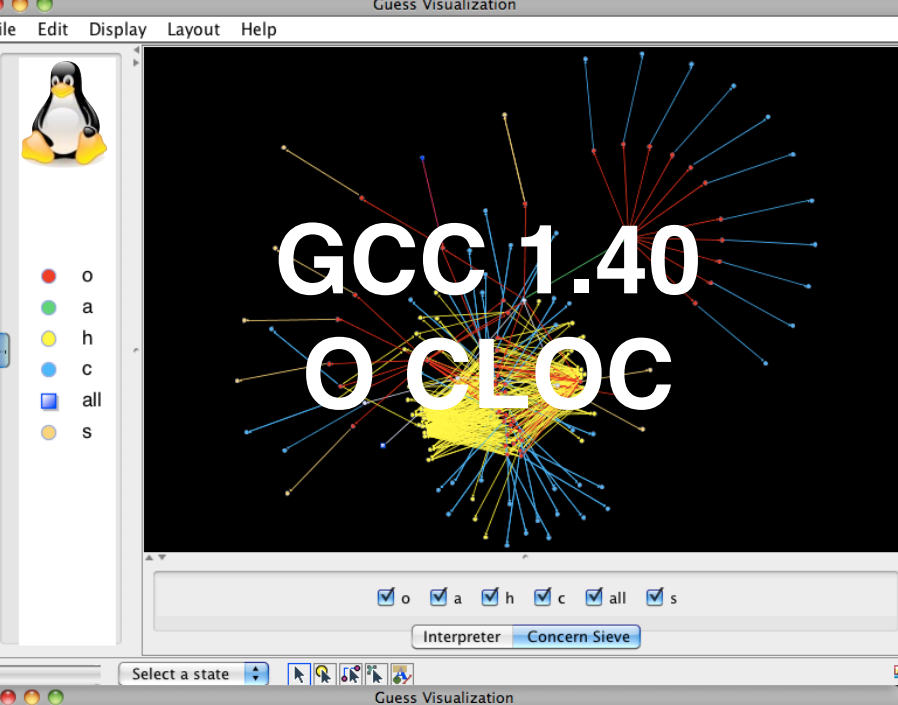
**Design recovery and
maintenance of build systems**

B. Adams *et al.*
[ICSM 2007]

**Tracing Software Build Processes
to Uncover License Compliance
Inconsistencies**

S. van der Berg *et al.*
[ASE 2014]







To summarize, these students should **always manually check** the systems they are studying, not just blindly run their scripts?



To summarize, these students should **always manually check** the systems they are studying, not just blindly run their scripts?

YES!!!



RELENG: International Workshop on Release Engineering

RELENG: International Workshop on Release Engineering

3 editions

230 participants

dozens of industry
& academic talks



RELENG: International Workshop on Release Engineering

3 editions

230 participants

dozens of industry
& academic talks



RELENG: International Workshop on Release Engineering

3 editions

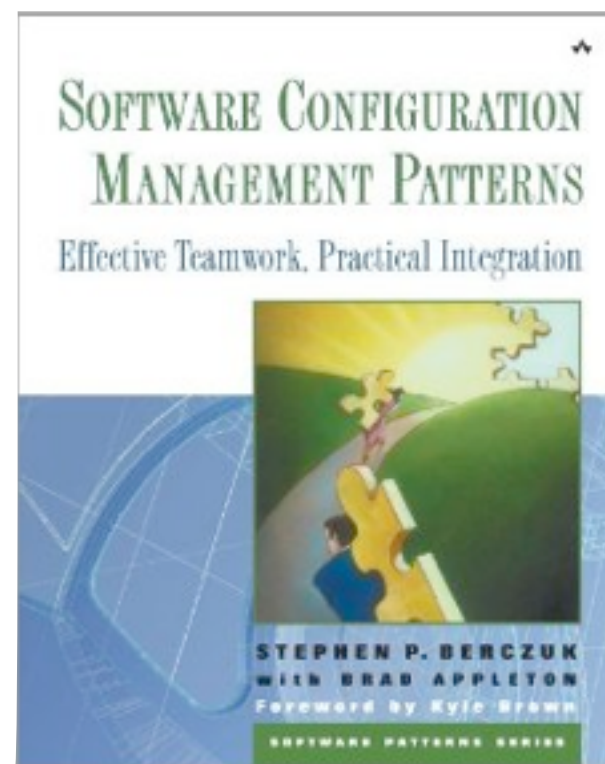
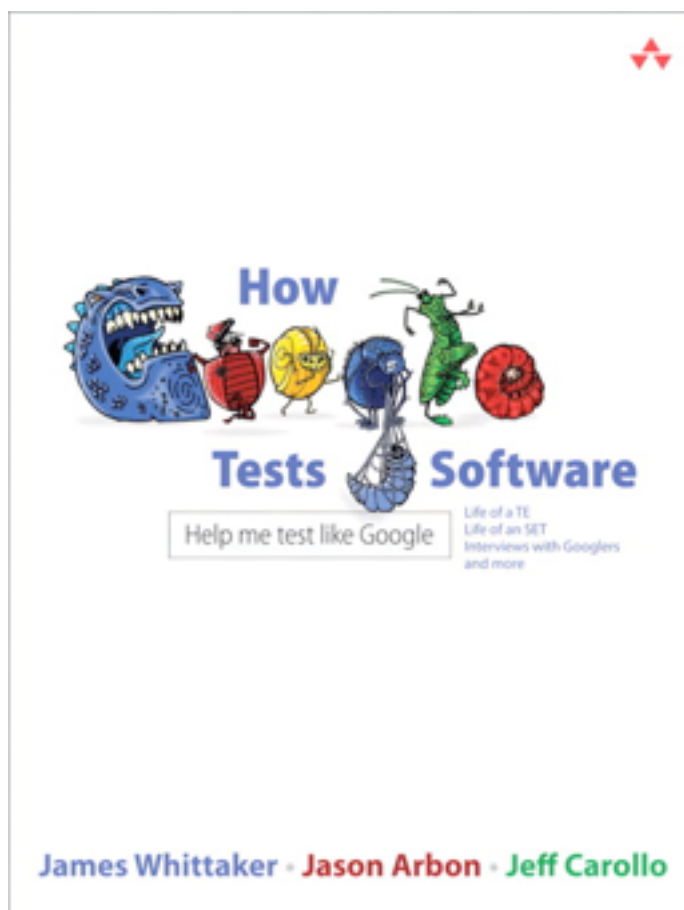
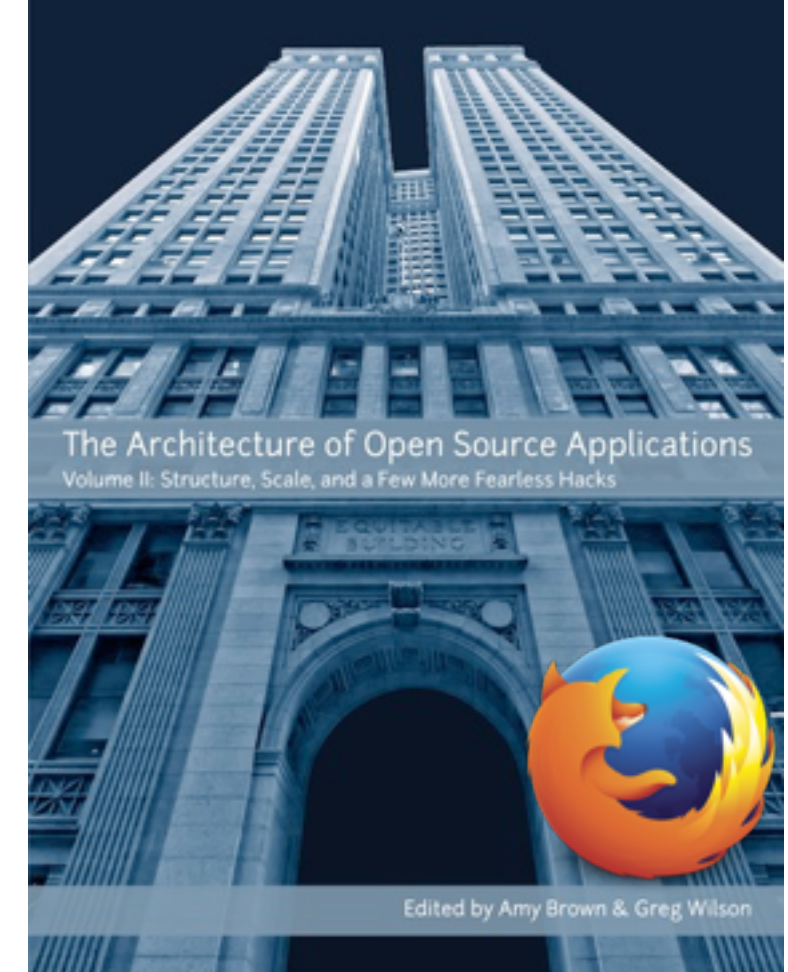
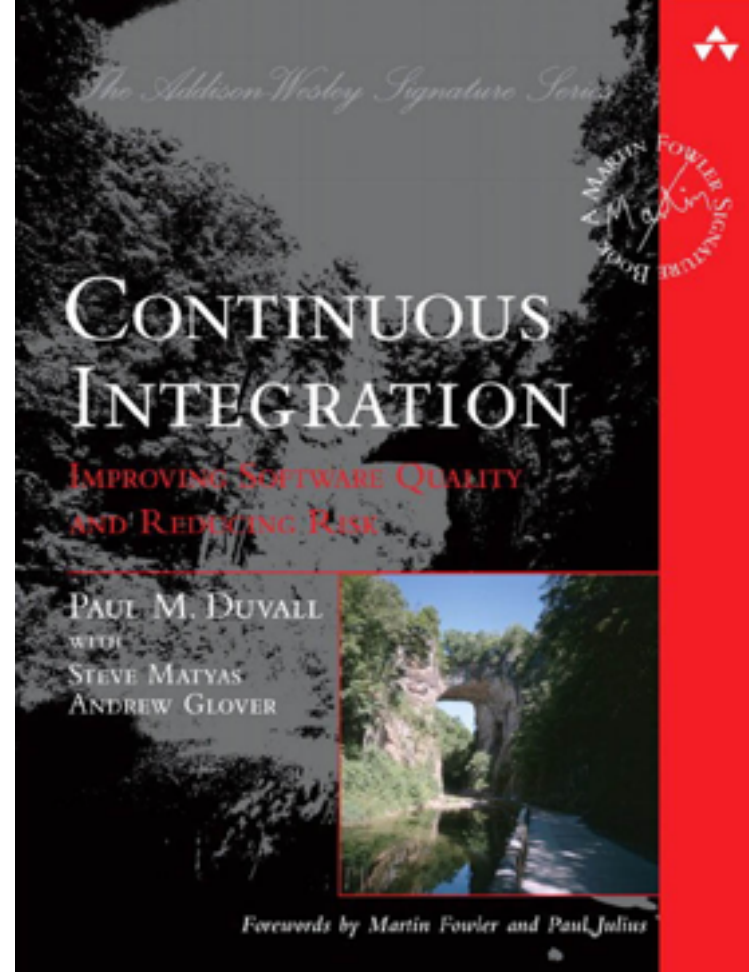
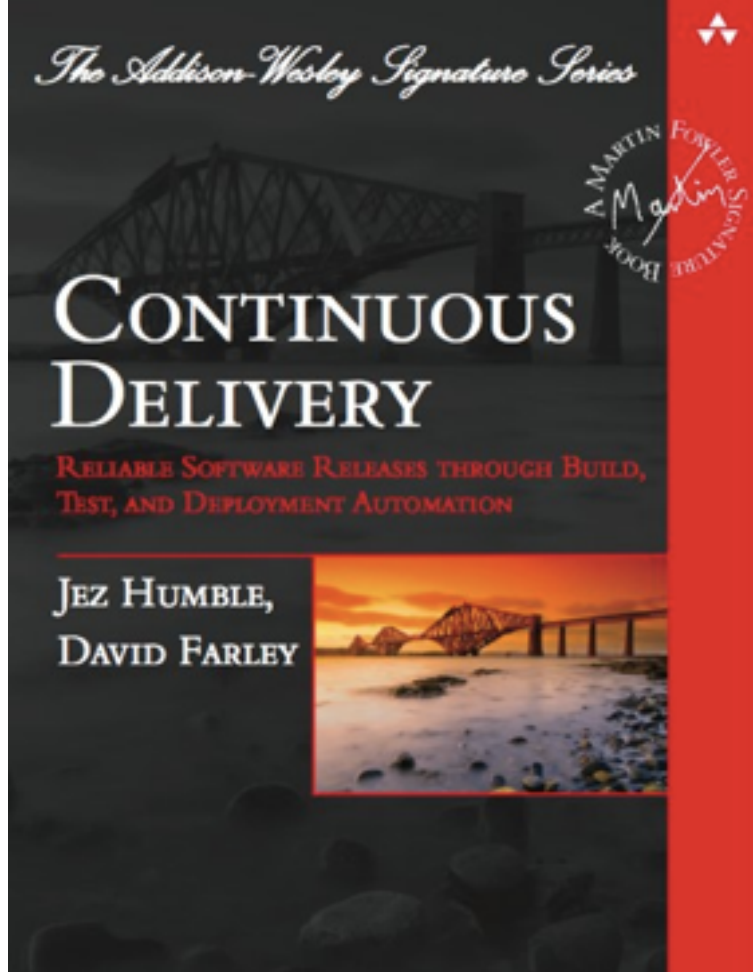
230 participants

dozens of industry
& academic talks



<http://releng.polymtl.ca>

RELENG 2016:
18/11 with FSE



<http://releng.polymtl.ca/RELENG2015/html/links.html>

<http://google-engtools.blogspot.ca/>



<http://www.openstack.org/blog/author/james-e-blair/>



integrating code changes



building/testing (CI)

Simplified Pipeline



releasing to the user



deploying a new release



integrating code changes



building/testing (CI)

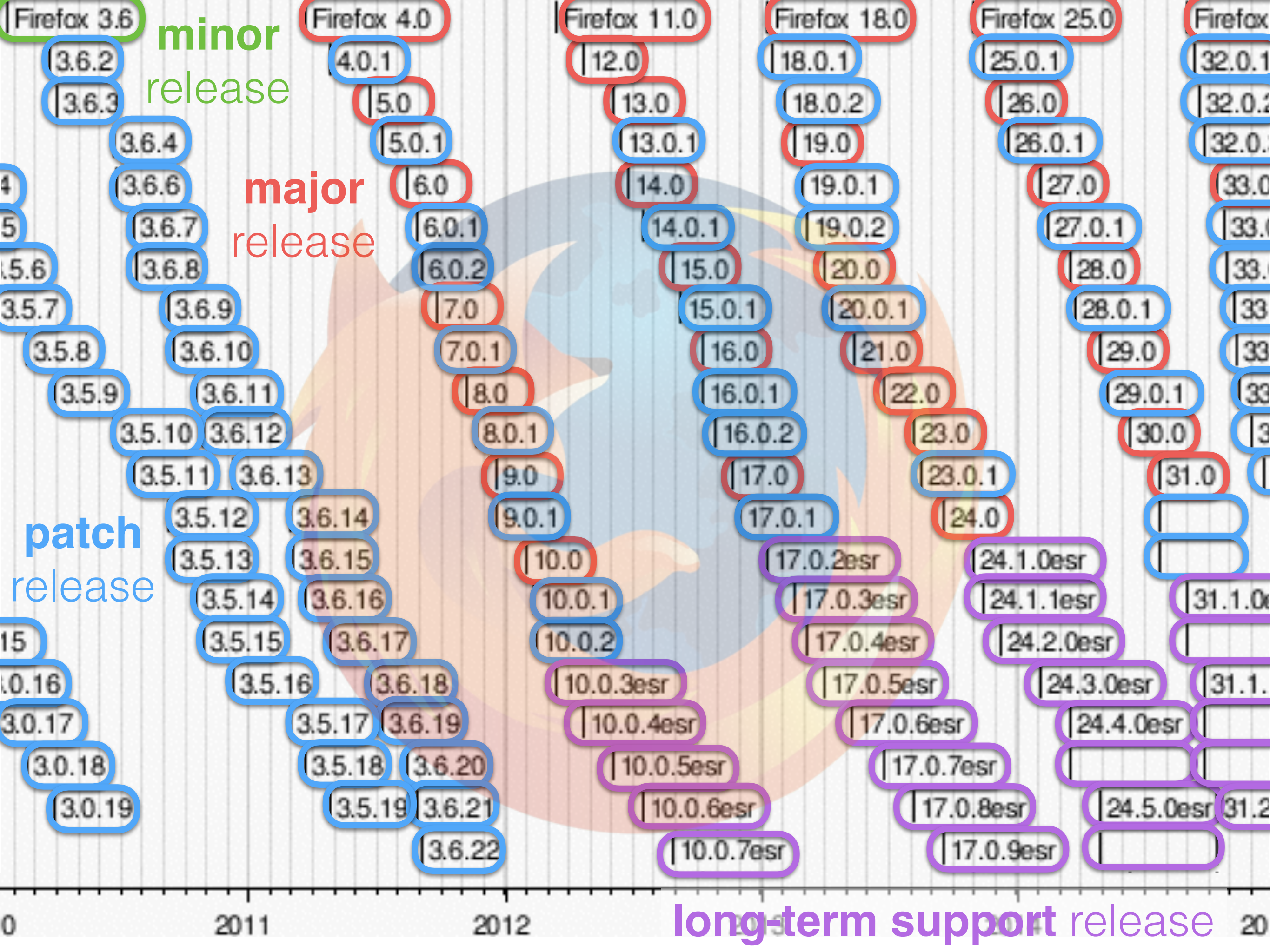
Simplified Pipeline

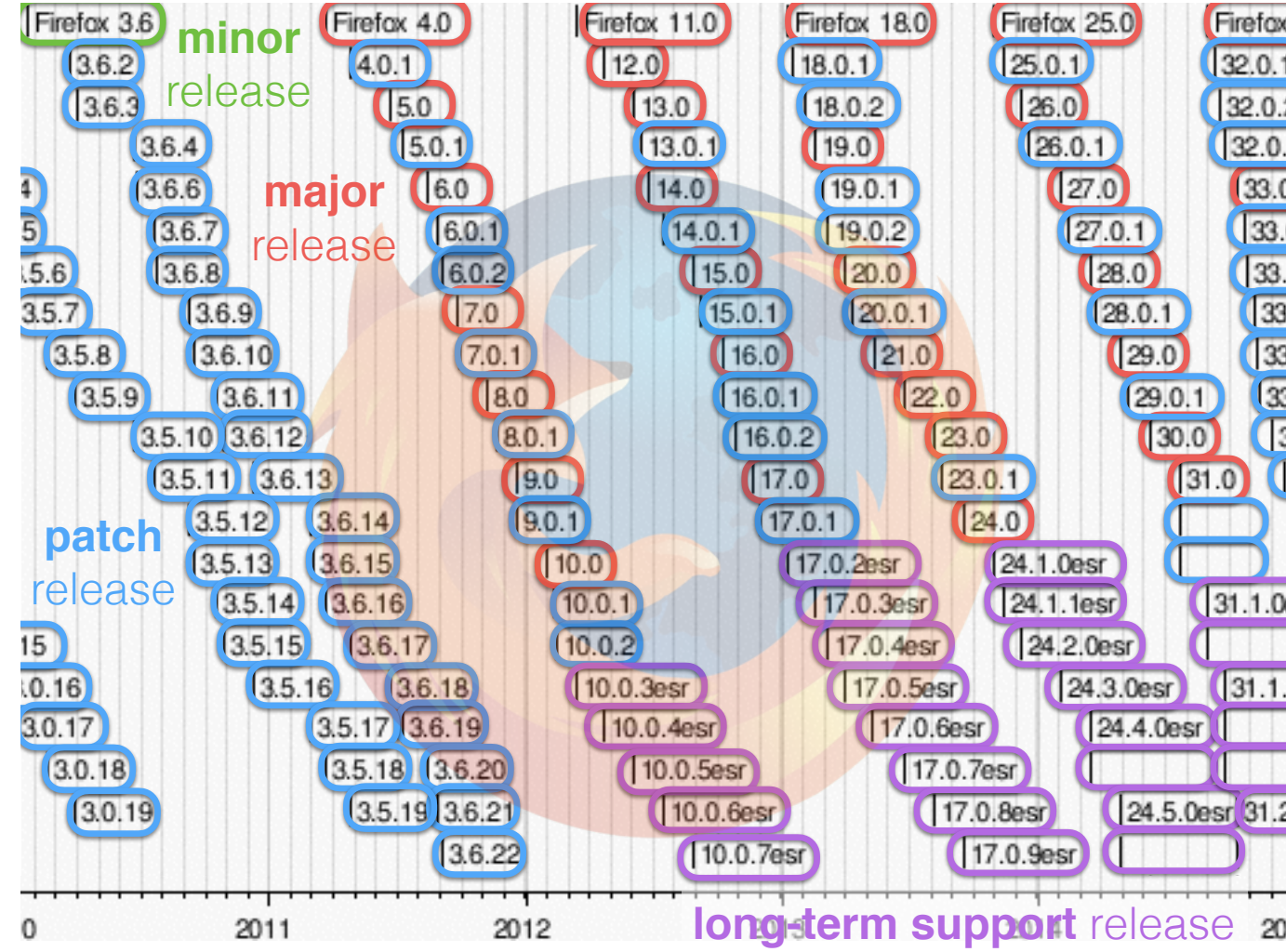
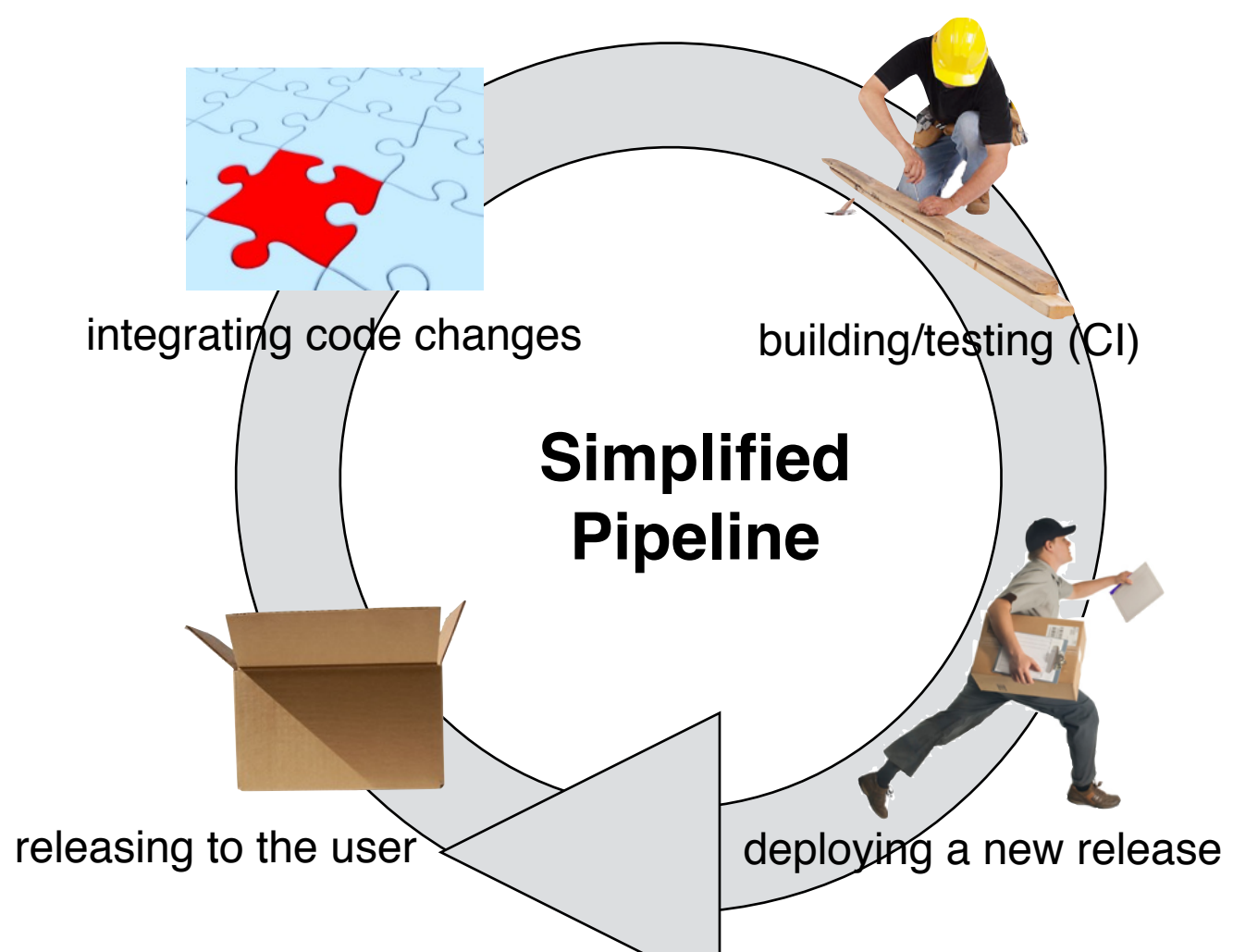


deploying a new release

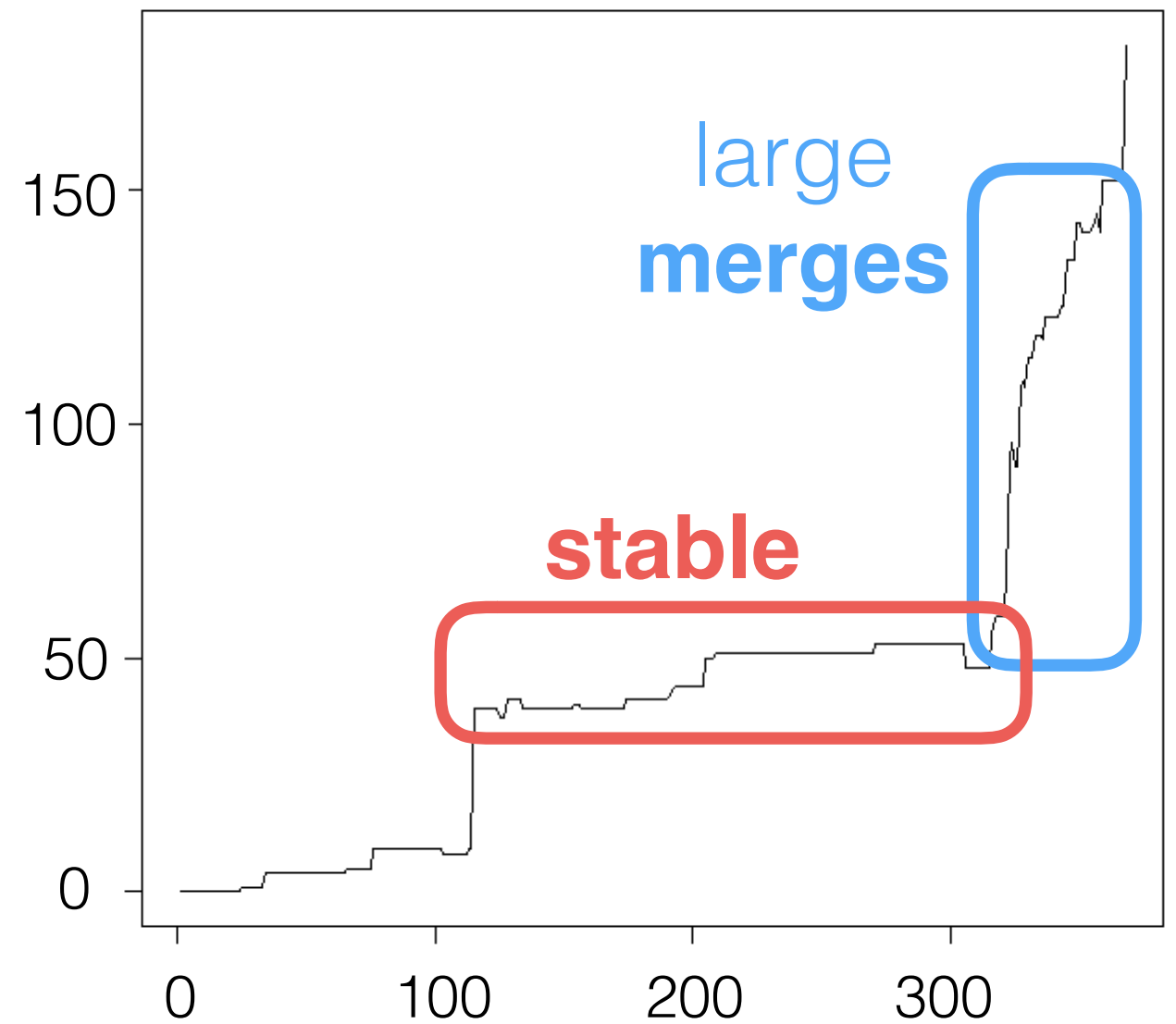
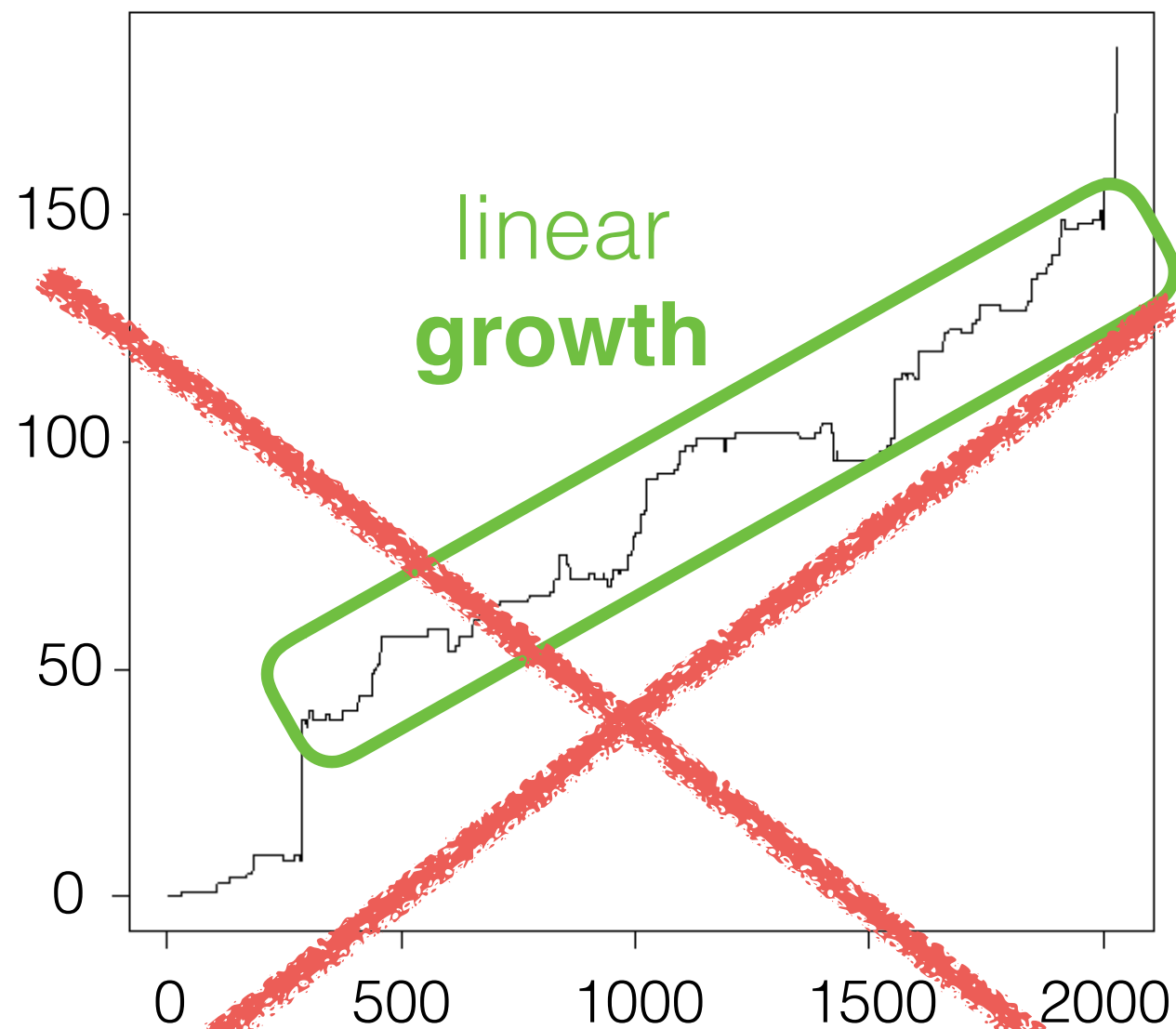


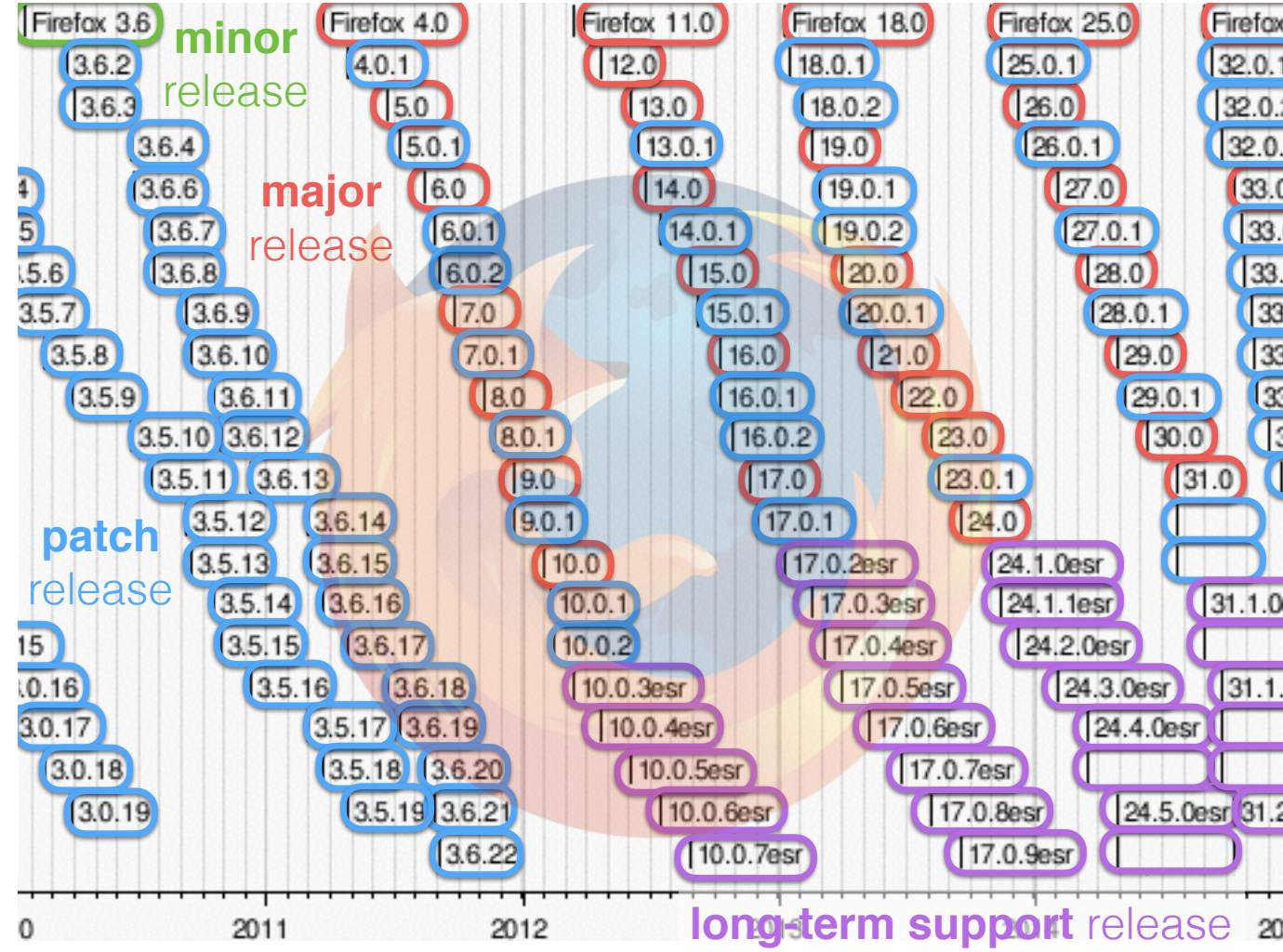
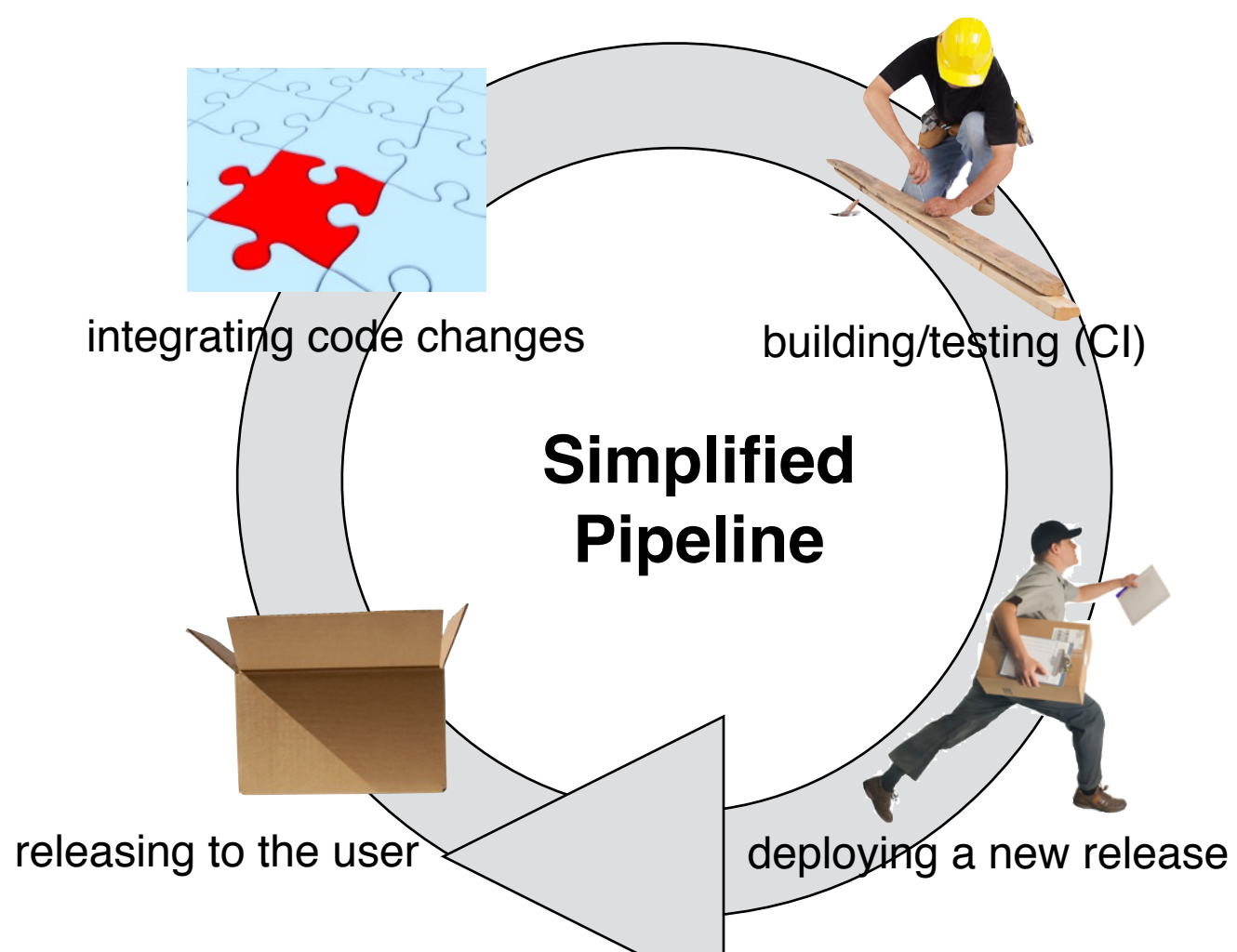
releasing to the user



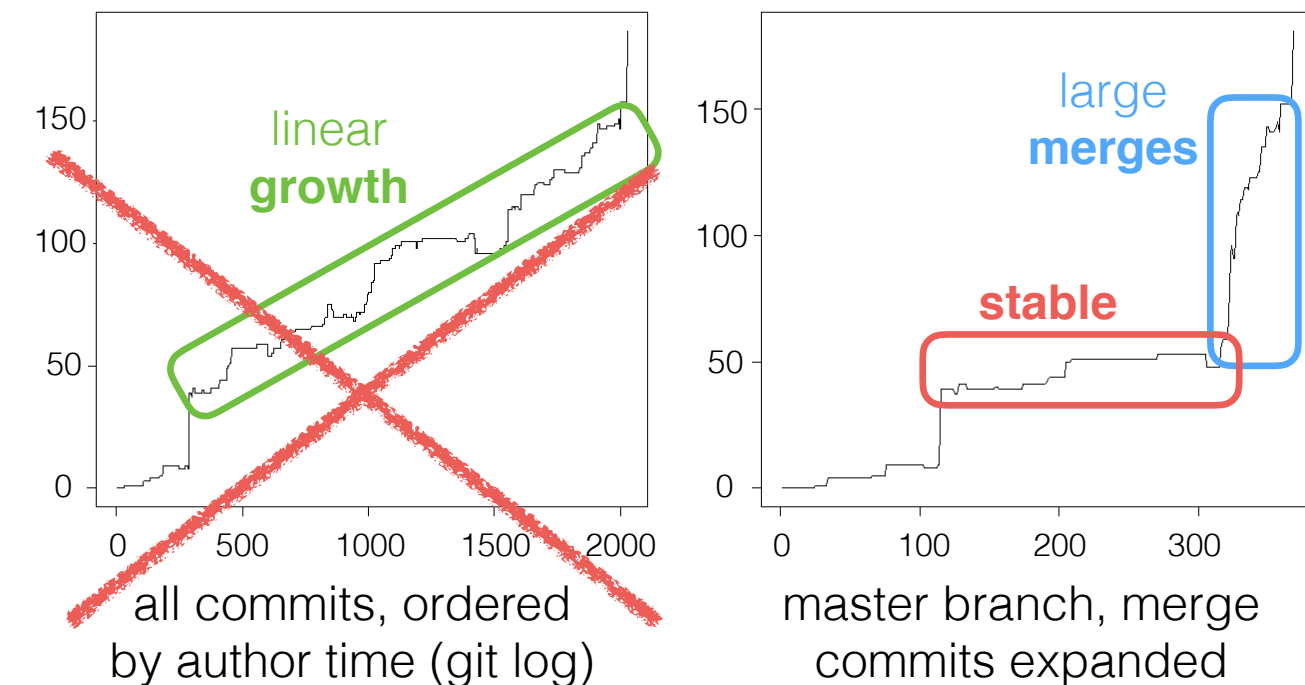


Evolution of #Files over Time

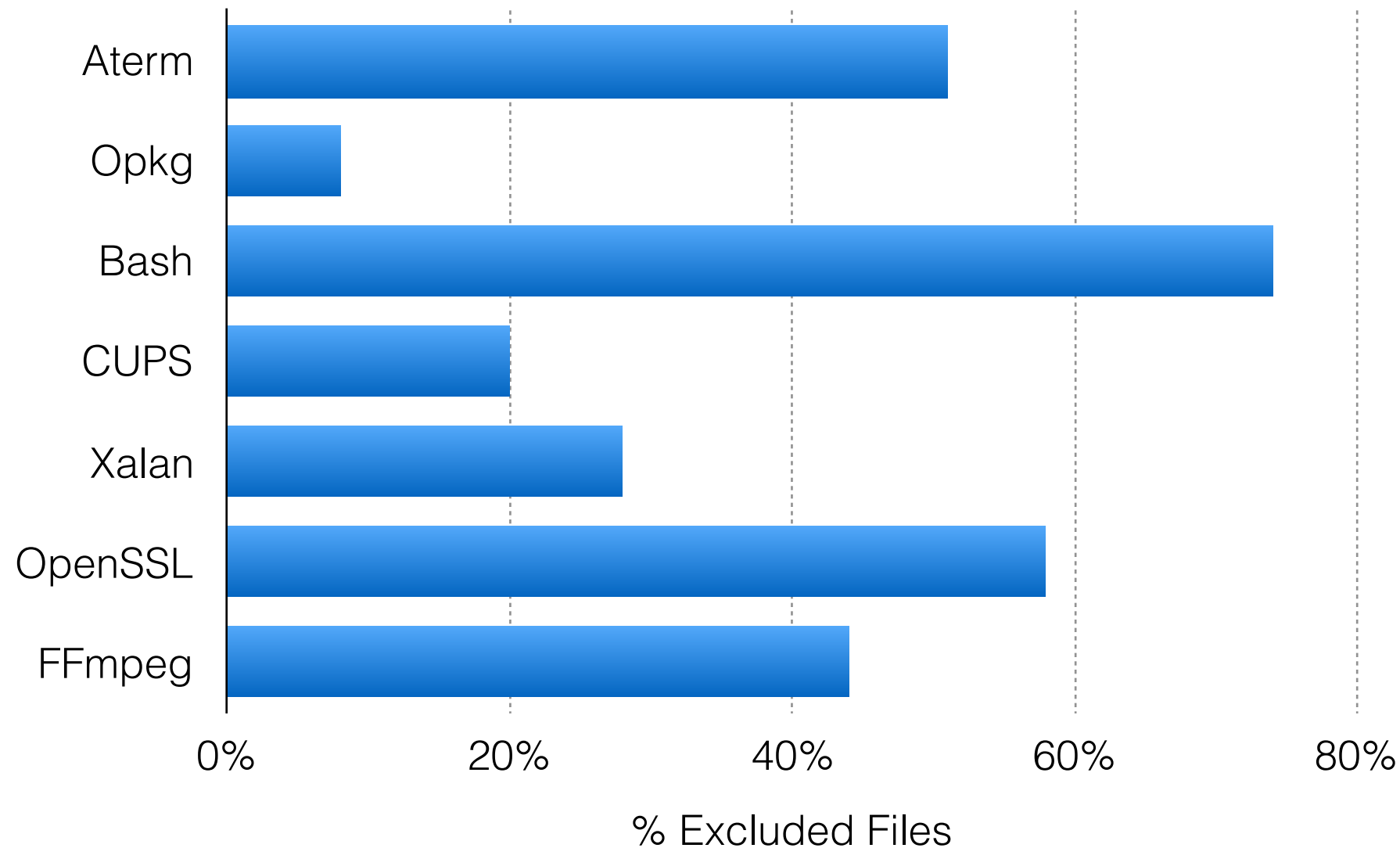




Evolution of #Files over Time



Many files are conditionally included in deliverables



Tracing Software Build Processes to Uncover License Compliance Inconsistencies

S. van der Berg *et al.*
[ASE 2014]

... because of **feature** selection, **operating system/hardware** configuration, **dead** code, ...

