Continuous Delivery In the Cloud

Ken Sykora - The Nerdery
@kensykora

Who are you?

The Nerdery

Principle Software Engineer

C# / ASP.Net Developer

Focus on DevOps / Cloud

Infrastructure/Automation Nut

World of Warcraft Junkie

Let’s talk about some terminology

Continuous Integration

Let’s define some terminology

Continuous Delivery

(what we’re talking about today)

Continuous Deployment

Continuous Deployment

Let’s define some terminology

Continuous Improvement

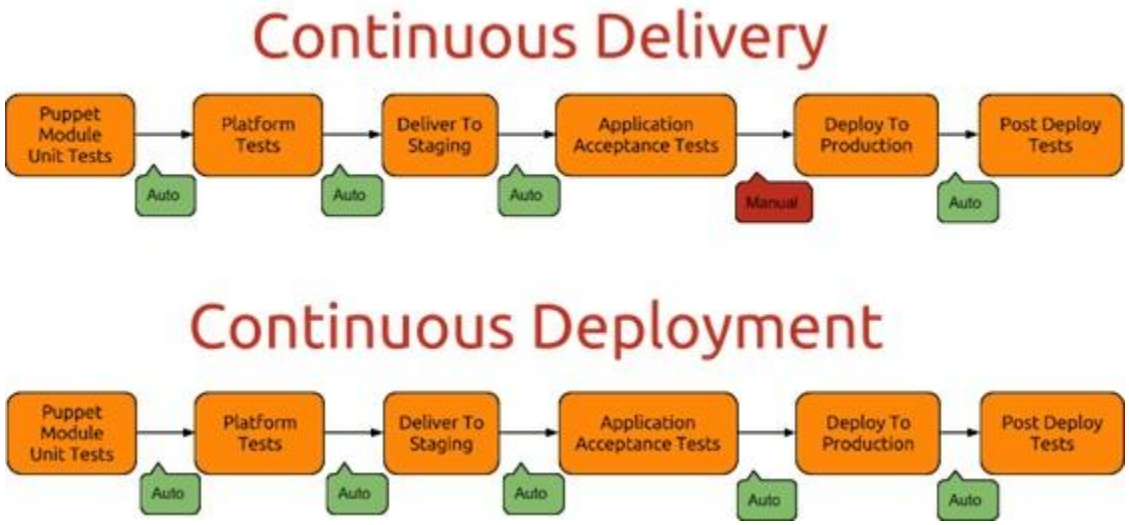
(what we should all aspire to incorporate)

Continuous Delivery

Each time I commit my code,
give me the confidence I need
that I could push that change to production today.

Continuous Delivery

Continuous Delivery vs Deployment



Credit: http://puppetlabs.com/blog/continuous-delivery-vs-continuous-deployment-whats-diff

The Build Pipeline

Source Code Application Development

The application is built and tested on local development environments. Developers make changes, add features, fix issues. Changes are checked into source control.

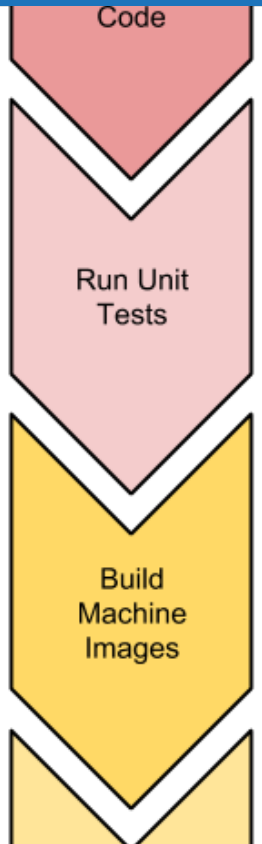
Build Application Code

The application is built and tested on a continuous integration server, which has similar capabilities to the developer's local workstation. At a minimum, the application must build correctly (compile, have all necessary resources in version control) or the application will fail when deployed.

Run Unit

Unit tests are responsible for testing the application code, at a functional level, does what it is expected to do. Given inputs yield expected outputs. If these do not pass, the application will not function correctly. Unit tests

The Build Pipeline



Unit tests are responsible for testing the application code, at a functional level, does what it is expected to do. Given inputs yield expected outputs. If these do not pass, the application will not function correctly. Unit tests can be run for both backend application code, and frontend javascript code.

If needed, servers that will run the application have their machine images built using provisioning tools. Machine images are built on the target deployment environment (e.g., Cloud Host, Virtualization Host)

Another set of tests can be run against the images created in the

The Build Pipeline

Verify and Validate Images

Another set of tests can be run against the images created in the previous step to validate that the proper actions were actually performed against them. A misconfigured instance can be difficult to diagnose, and additional checks prior to production deployment reduce this risk.

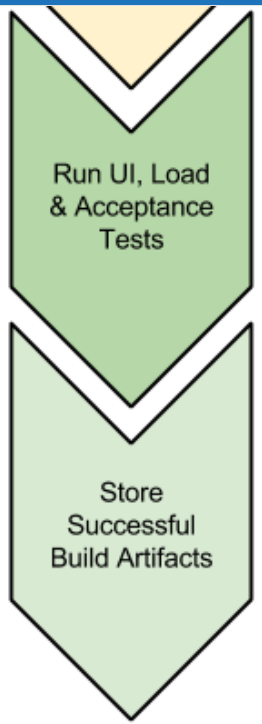
Deploy Provisional System

A provisional system is spun up based on the current build, using the assets, binaries, and machine images from the previous build stages that would reflect a system similar to a production environment. This provisional system will be used for running further tests.

Run UI, Load & Acceptance

Automated UI Tests are run on all supported browsers to verify all required test cases pass. Load testing is run against the provisional environment to ensure the amount of expected traffic can be handled.

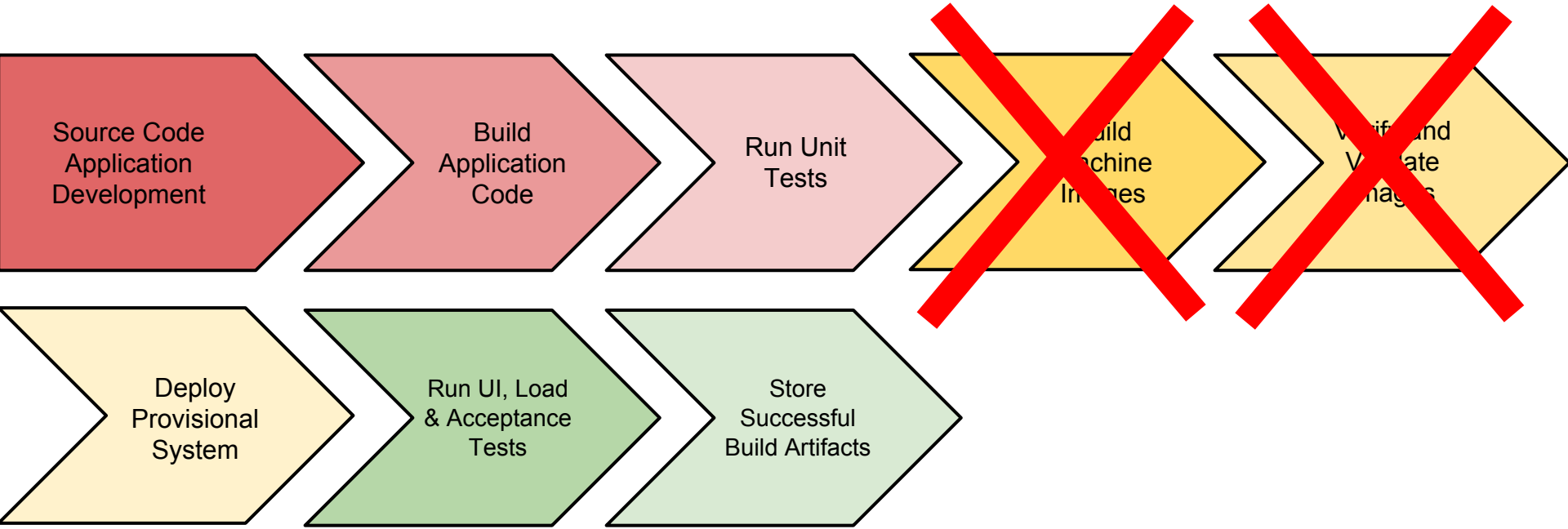
The Build Pipeline



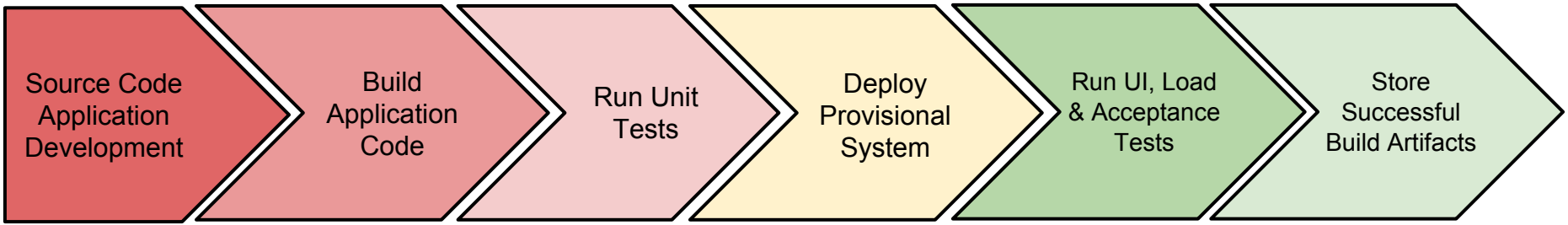
Automated UI Tests are run on all supported browsers to verify all required test cases pass. Load testing is run against the provisional environment to ensure the amount of expected traffic can be handled. Once these tests are completed, the provisional environment is then shut down.

Application code is stored, and machine images are tagged in virtualization environment, while stale images and artifacts are cleaned up. Build is flagged as a success and a script can now be run to deploy the environment and switch over traffic to the new production build, and take down the old production servers.

What we'll be looking at

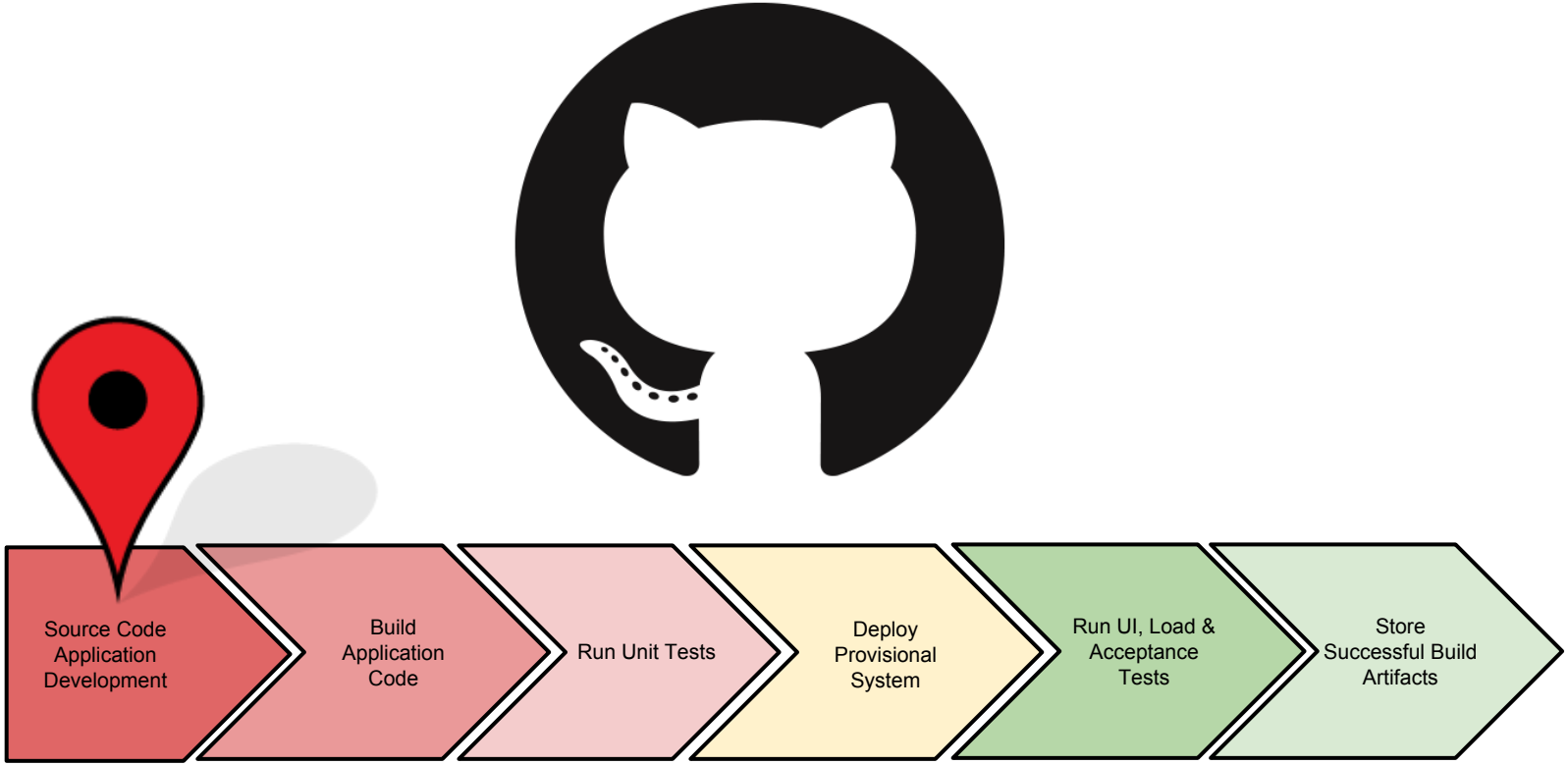


What we'll be looking at



Cloud Tools

Source / Application Development



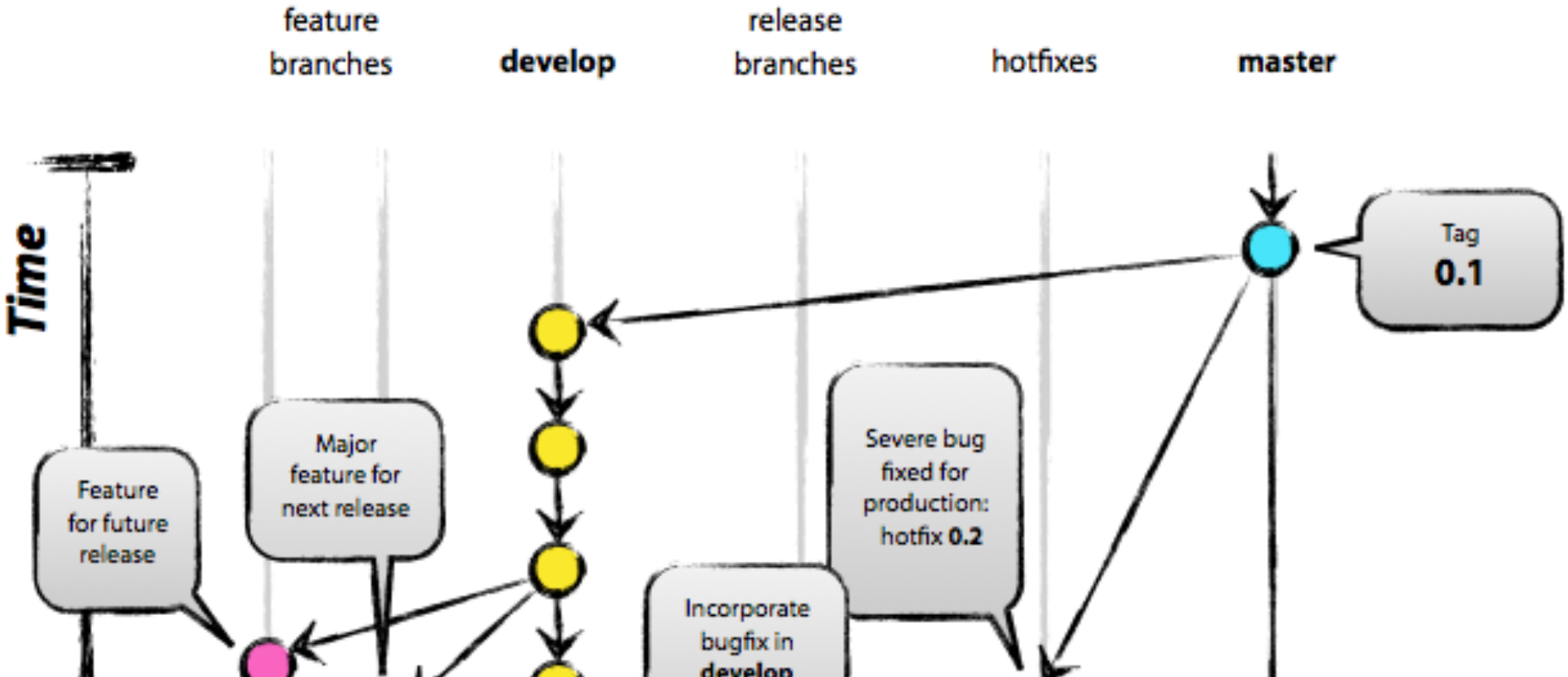
Source Code / App Versioning

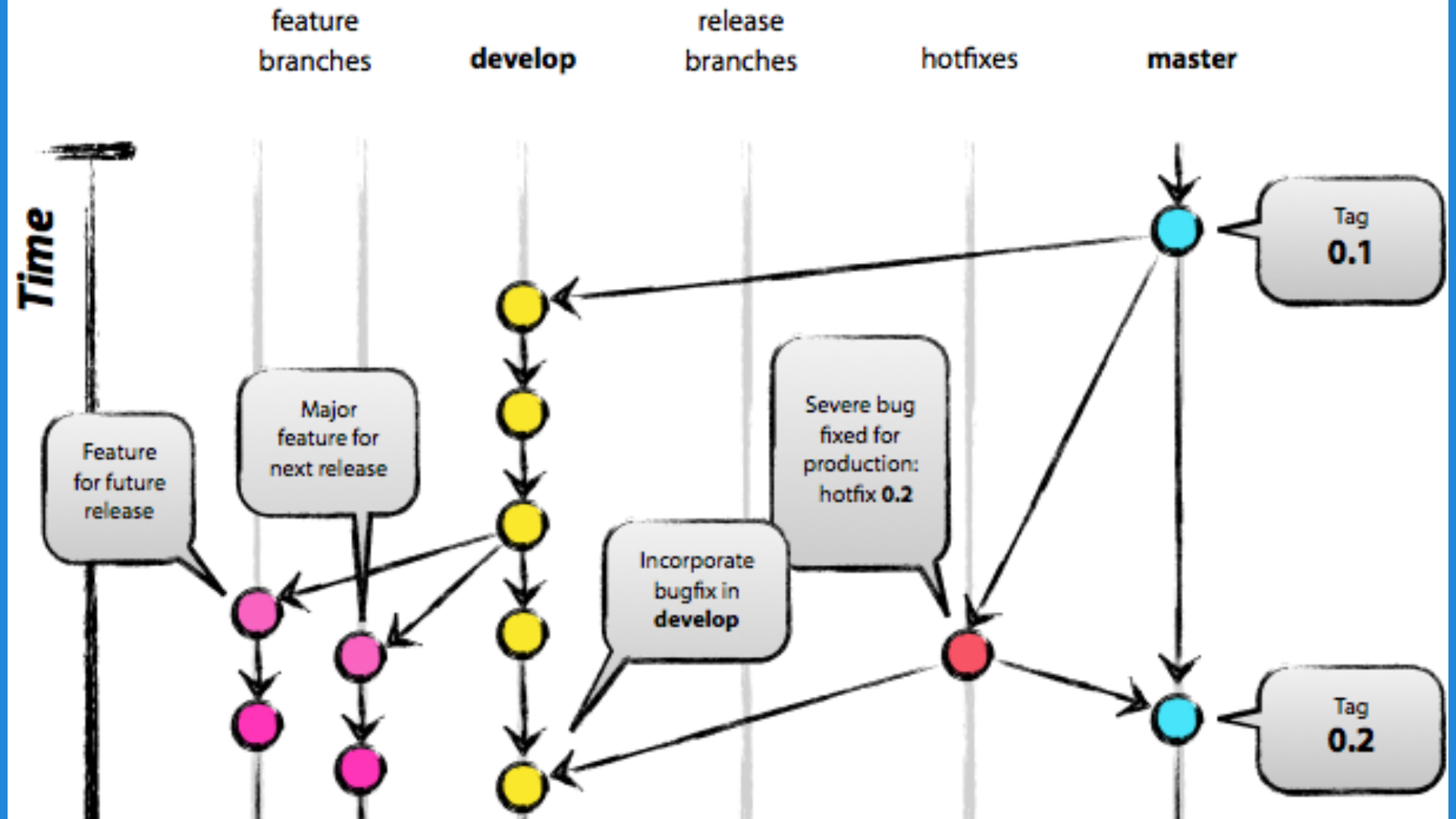
What release are we on?

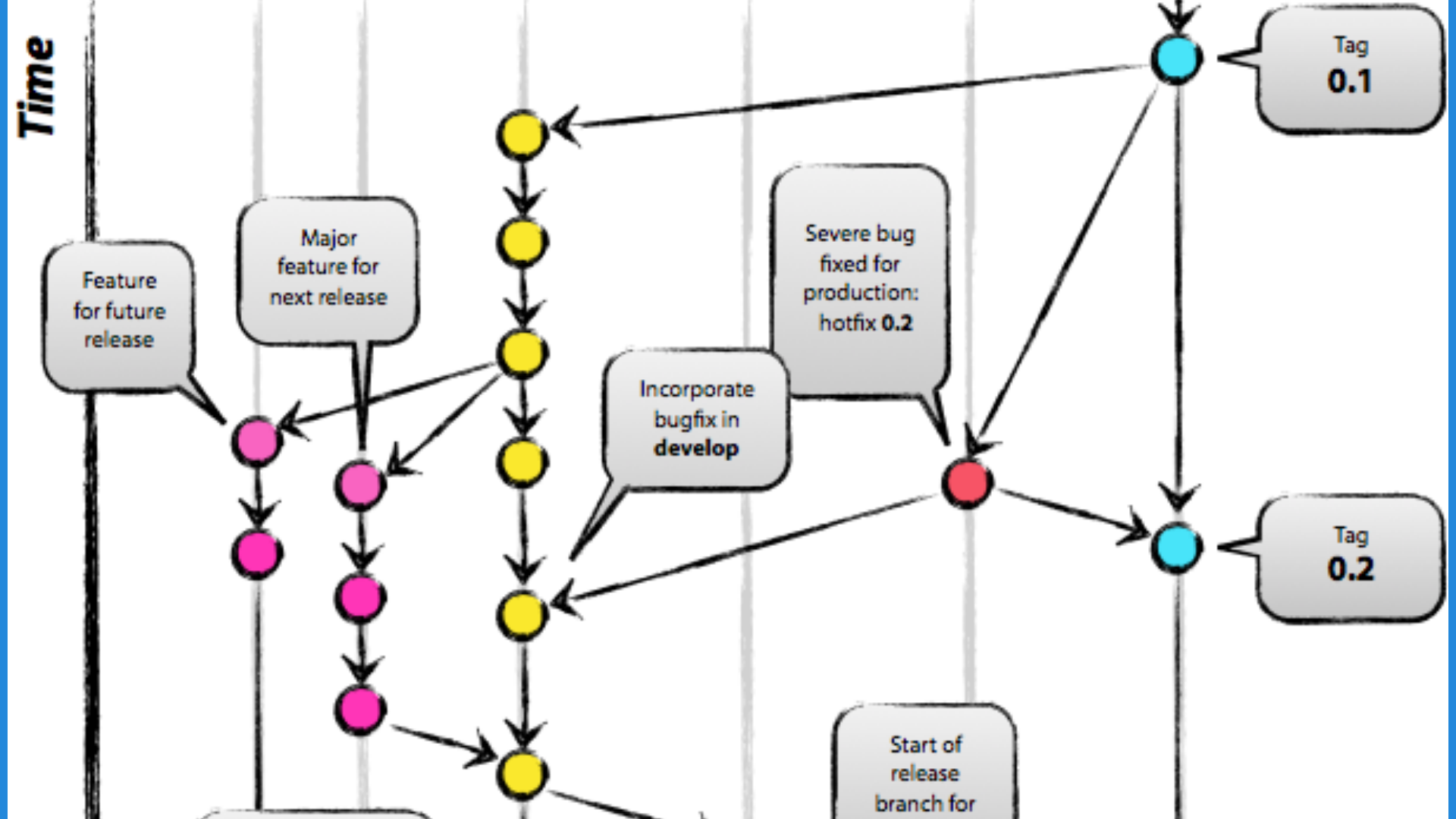
What's the current version out there?

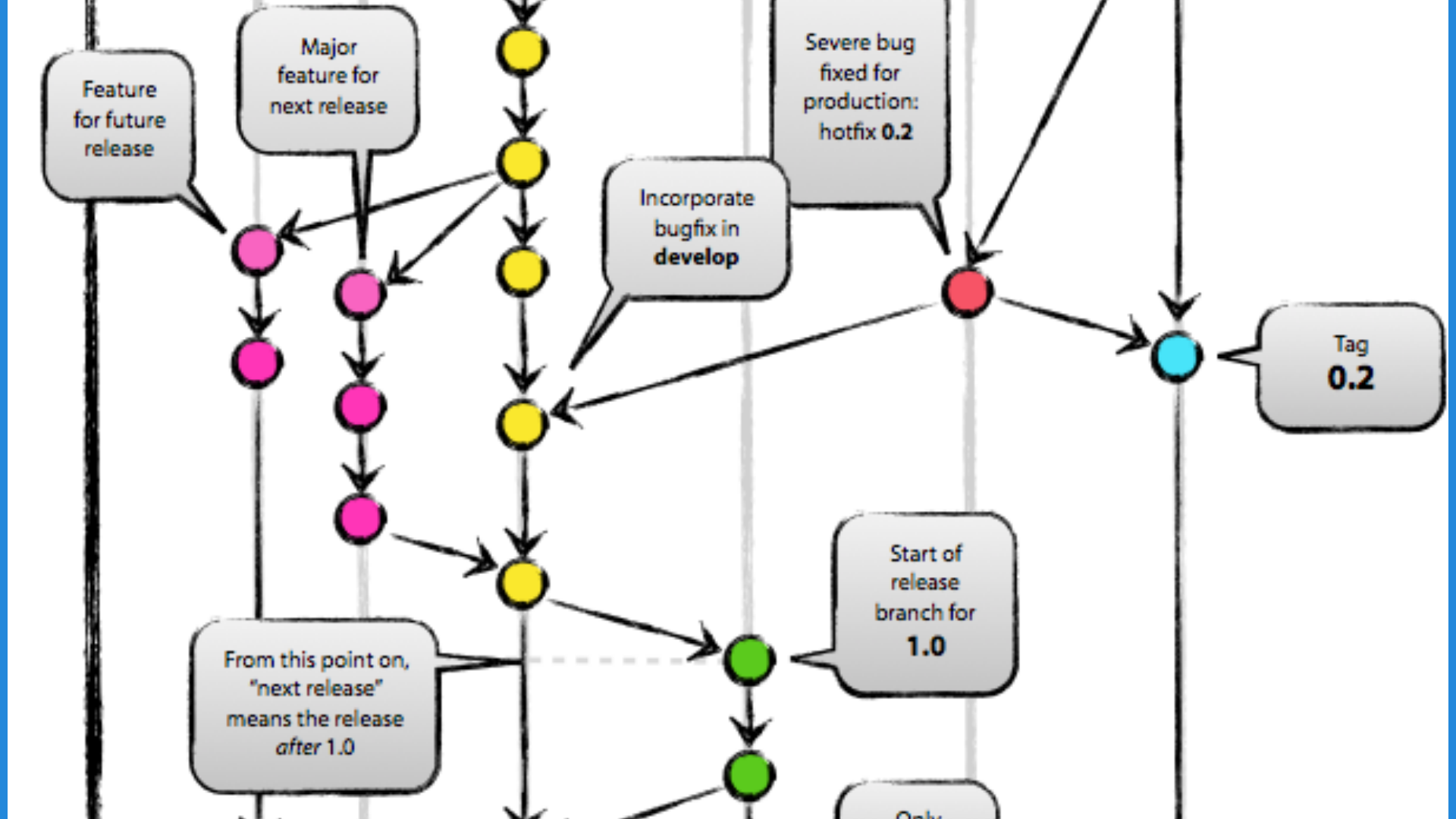
What is being developed and what can we deploy?

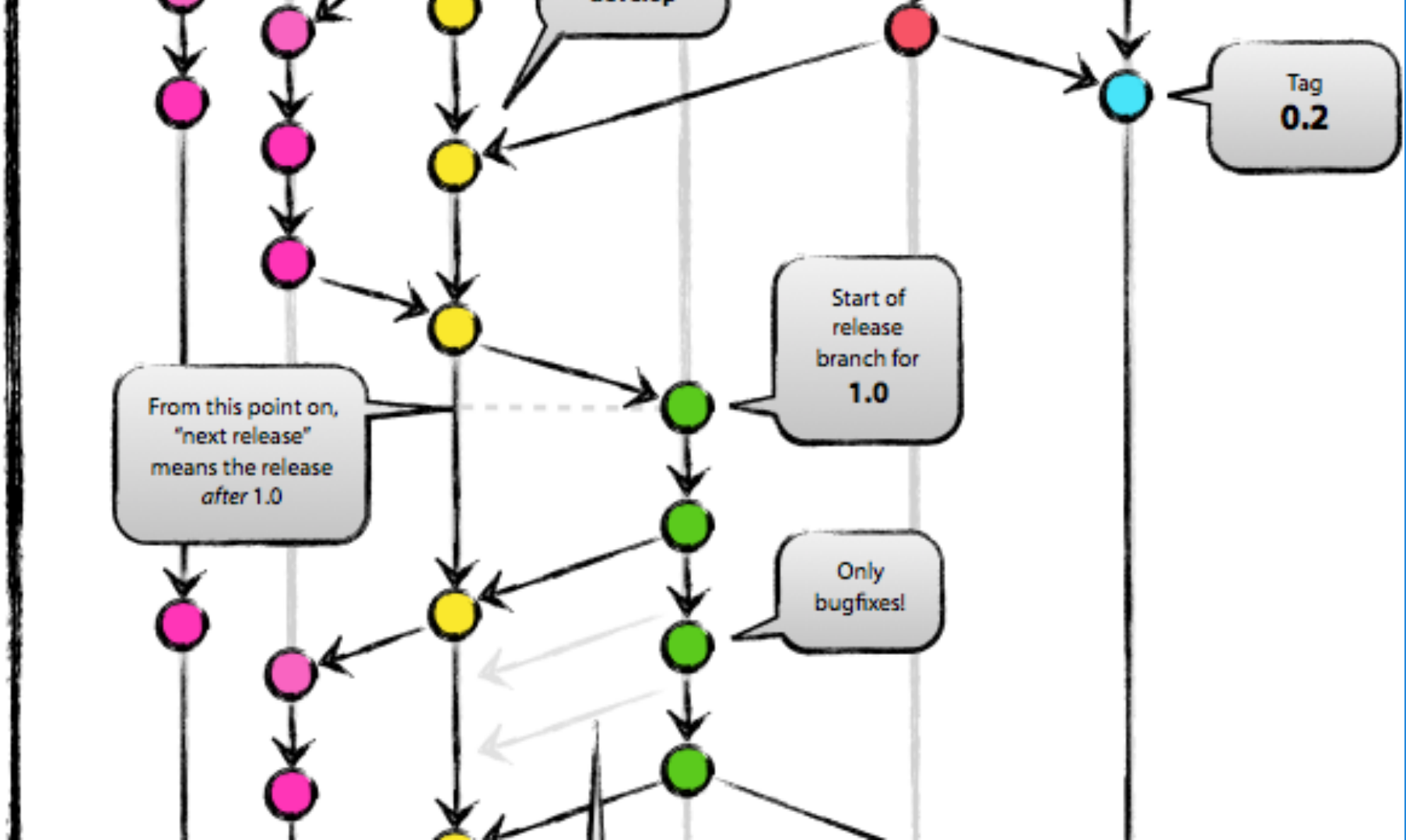
Git Flow (courtesy Vincent Driessen)

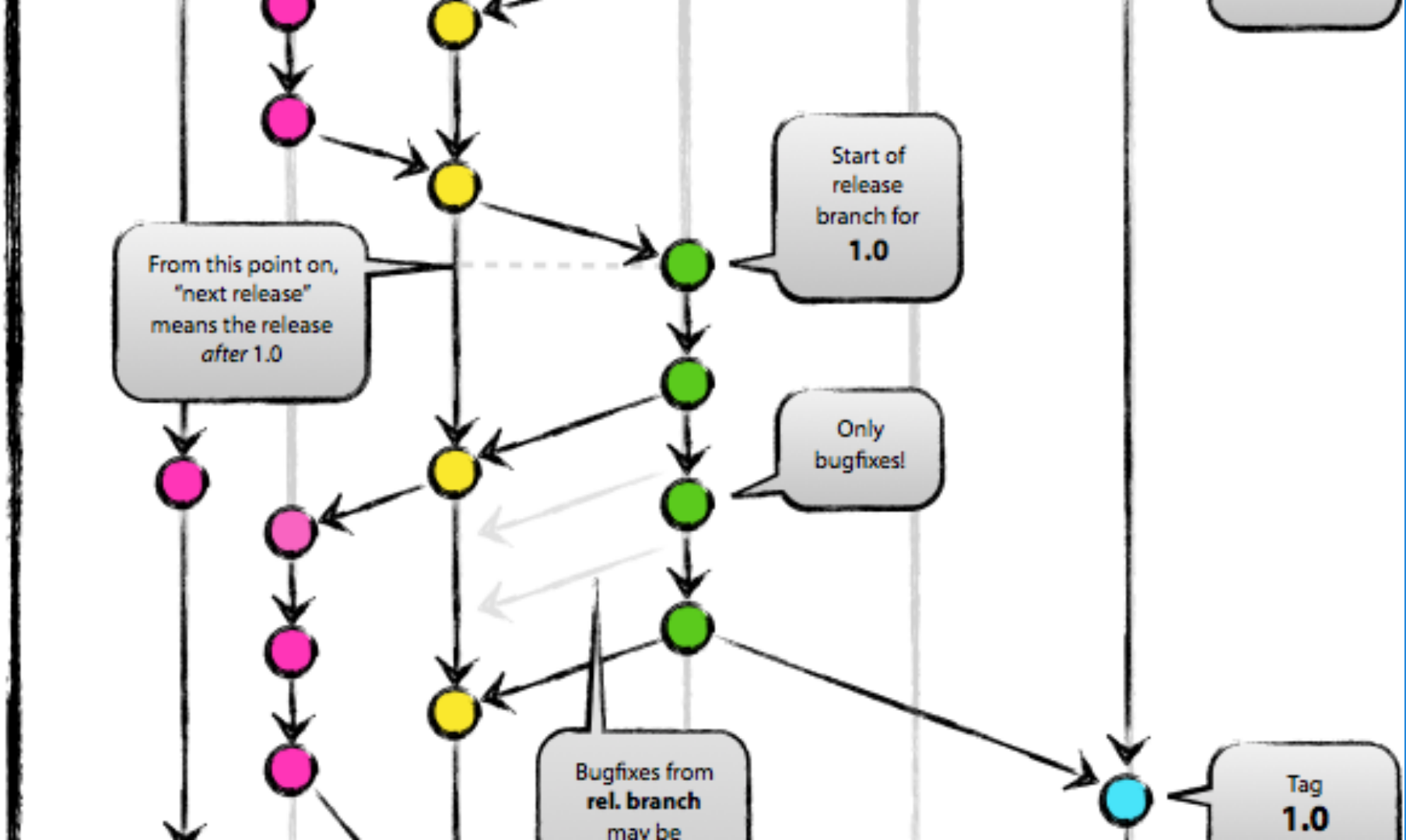


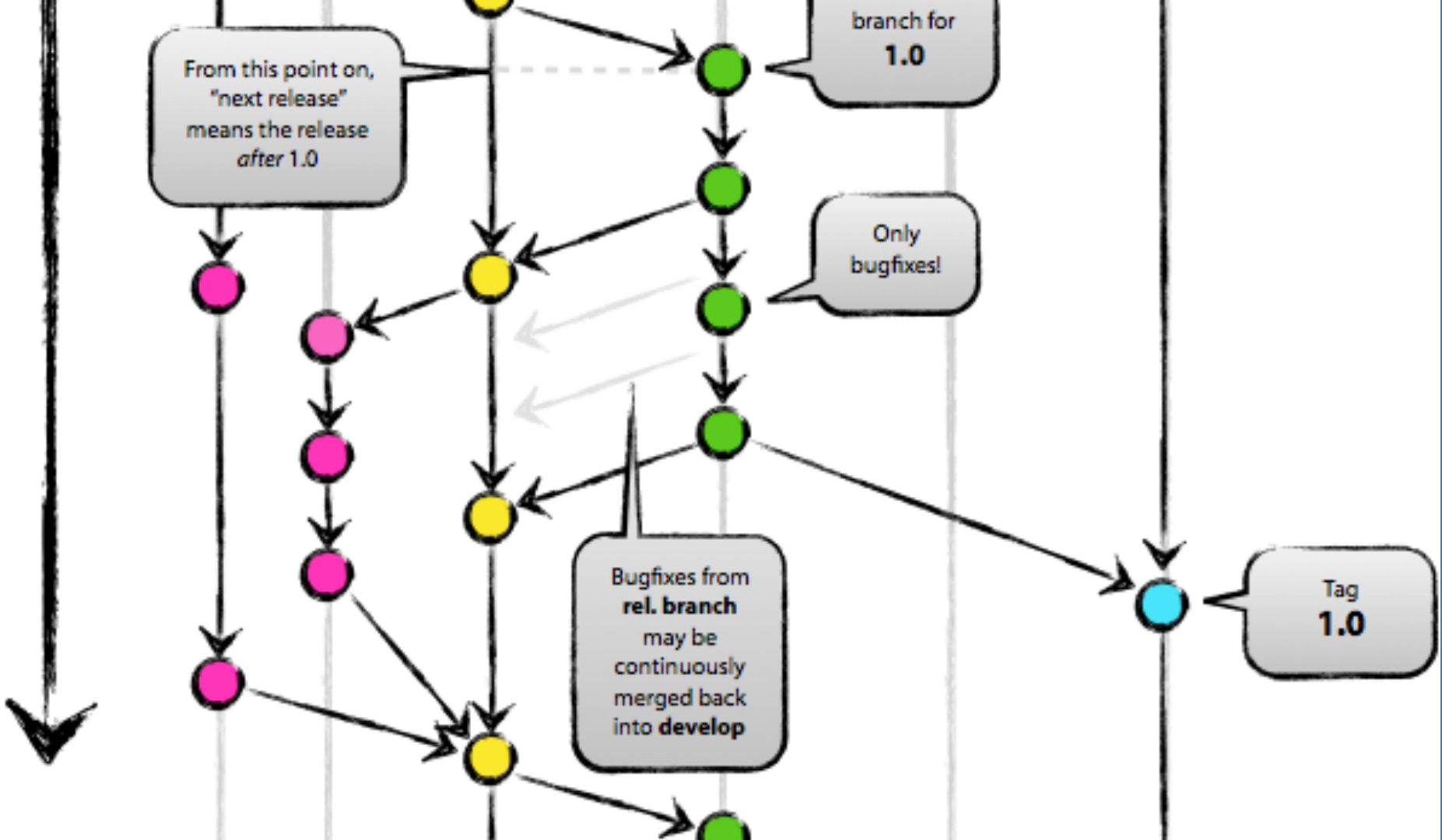

feature branches
develop
release branches
hotfixes
master
Time
Feature for future release
Major feature for next release
Incorporate bugfix in develop
Severe bug fixed for production: hotfix 0.2
Tag 0.1
Tag 0.2

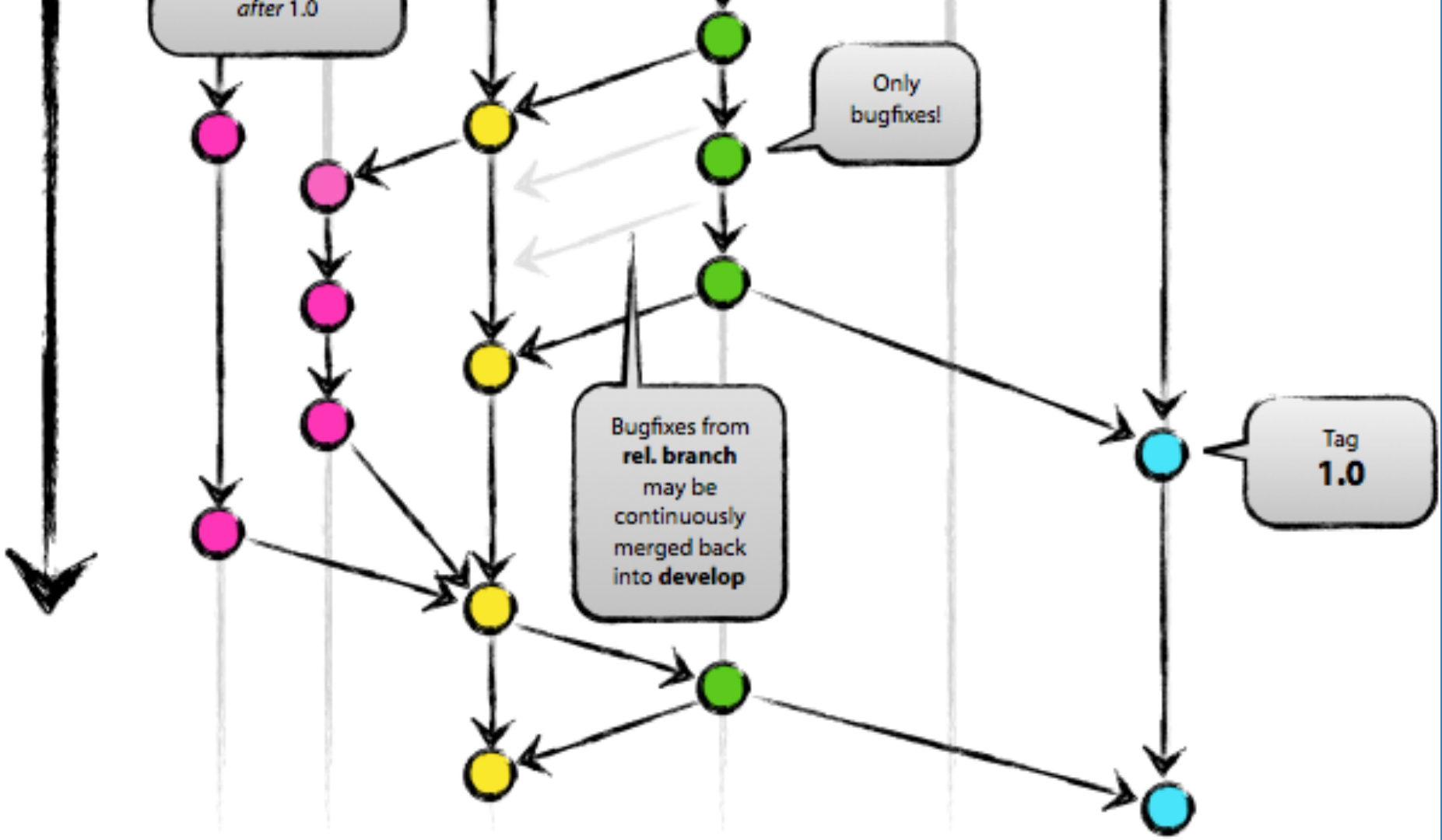

Time
Tag
0.1
Feature
for future
release
Major
feature for
next release
Severe bug
fixed for
production:
hotfix 0.2
Incorporate
bugfix in
develop
Tag
0.2
Start of
release
branch for


Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix **0.2**
Incorporate bugfix in **develop**
Tag **0.2**
Start of release branch for **1.0**
From this point on, "next release" means the release *after* 1.0


Tag
0.2
Start of
release
branch for
1.0
From this point on,
"next release"
means the release
after 1.0
Only
bugfixes!


Start of release branch for 1.0
From this point on, "next release" means the release *after* 1.0
Only bugfixes!
Bugfixes from **rel. branch** may be
Tag **1.0**


branch for
1.0
From this point on,
"next release"
means the release
after 1.0
Only
bugfixes!
Bugfixes from
rel. branch
may be
continuously
merged back
into **develop**
Tag
1.0

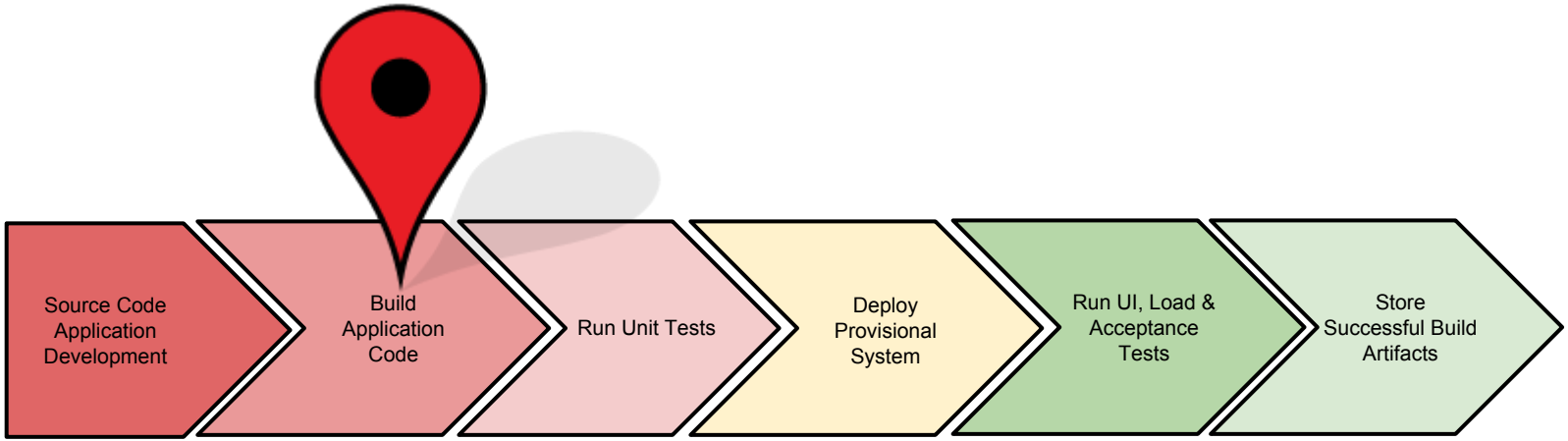

after 1.0
Only bugfixes!
Bugfixes from **rel. branch** may be continuously merged back into **develop**
Tag **1.0**

Including Version in Website

MSBuild Community Tasks

--Assembly Patching

--Environment Settings



A slide on environment variables

http://ss64.com/nt/set.html

https://www.digitalocean.com/community/tutorials/how-to-read-and-set-environmental-and-shell-variables-on-a-linux-vps

Another slide on environment variables

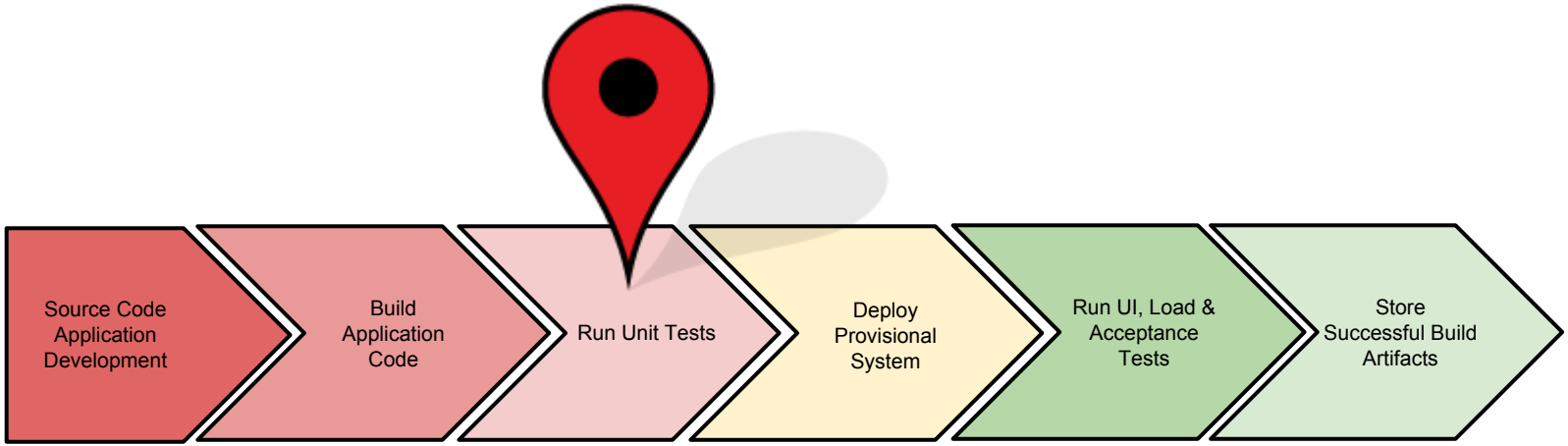
Uses for Environment Variables:

- Sensitive Information
  - API Keys / Passwords
  - Connection Strings
- Controlling Build Flow
  - Flags on/off
  - Build Configurations
- Environment Specific Details
- Build Specific Details
  - Build #
  - Source Repository Information

Versioning - The Final Result

Demo

Testing


Source Code Application Development
Build Application Code
Run Unit Tests
Deploy Provisional System
Run UI, Load & Acceptance Tests
Store Successful Build Artifacts

Testing

Test Types:

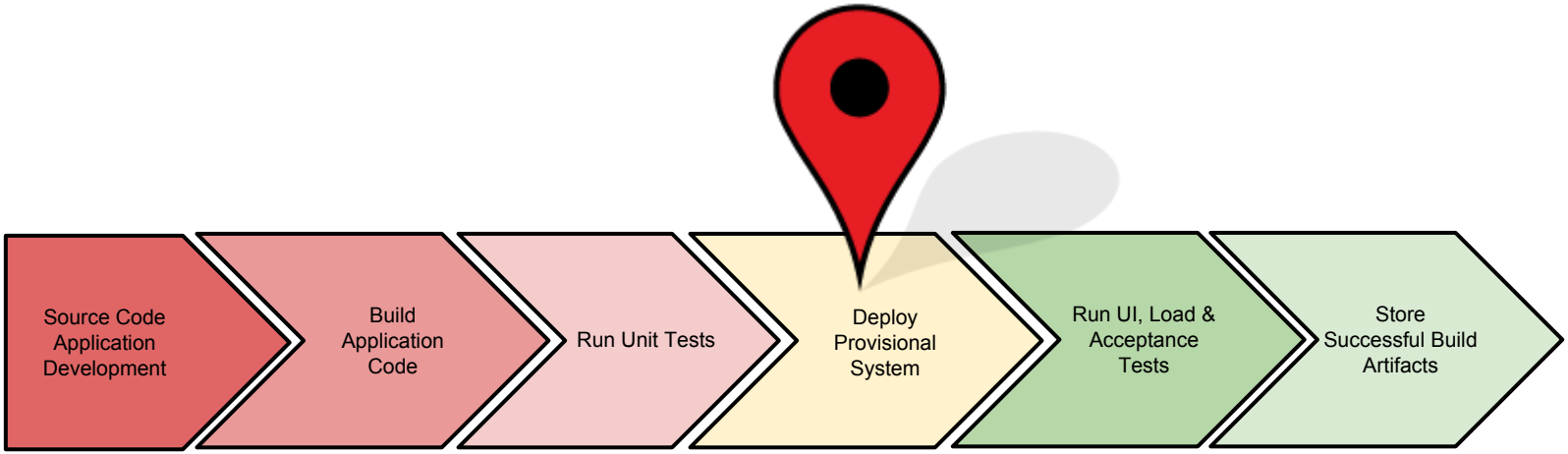
- Unit Tests (Services / MVC Controllers)
- Full Stack Tests
- Integration Tests
- UI Tests (Selenium)
- Load Tests
- User Acceptance Tests

Unit Tests

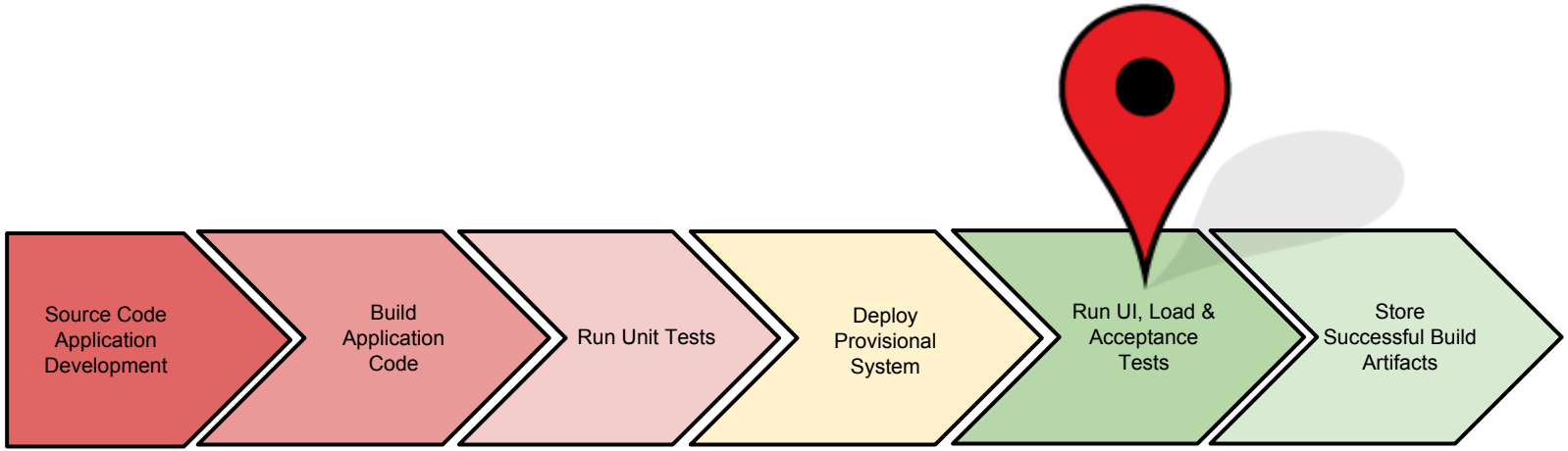
Demo

Full Stack Testing

- Setup mini-http server using C# unit tests
- Deploy to temporary site for the purpose of automated testing

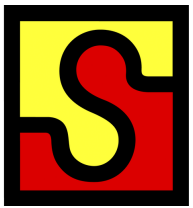


Additional Testing



UI Testing - Selenium

- Test Framework
- Supports most unit test frameworks
- Supports multiple languages
- Supports multiple browsers
- PaaS offerings built up around it



UI Tests - Why?

- Faster time to failure
- Cross-Browser / Cross-Platform
- Value in repeatability (Regression Testing)

Testing

Demo

UI Tests - Why Not?

Drawbacks:

- View Specific
- Time consuming to create

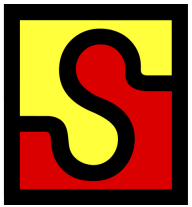
When Not to Use:

- Microsites
- Sites with short lifetimes
- “One-and-done”

OMGGGGG SHUT UP ABOUT TESTING LET'S SEE THE CLOUD STUFFFFFFFFF

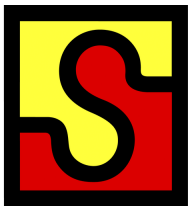
Sauce Labs

- PaaS Testing environment for running your selenium tests
- Can run against public sites, or use SauceConnect to reach internal servers



Sauce Labs

Demo



Load Testing

- Sending a ton of traffic to your site to ensure it can hold it’s own.
- Can be extremely expensive
- Options out there to do a lightweight version of a true load test

Load Testing - Loader.io

DEMO

Continuous Integration - AppVeyor

- PaaS Build Server based on Windows / .Net
- OOTB no-Configuration builds, testing, and artifacting
- Source control driven configuration

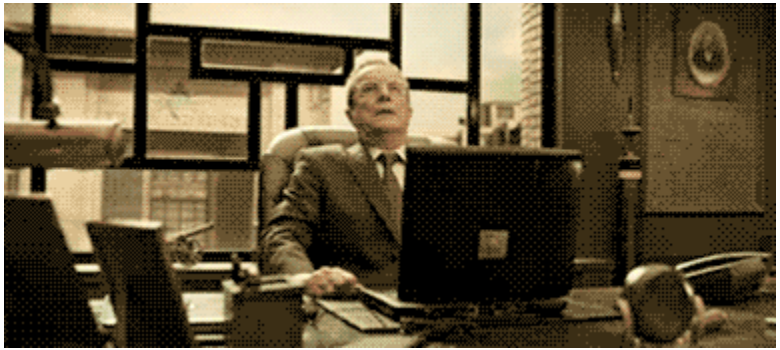
AppVeyor - appveyor.yml

Demo

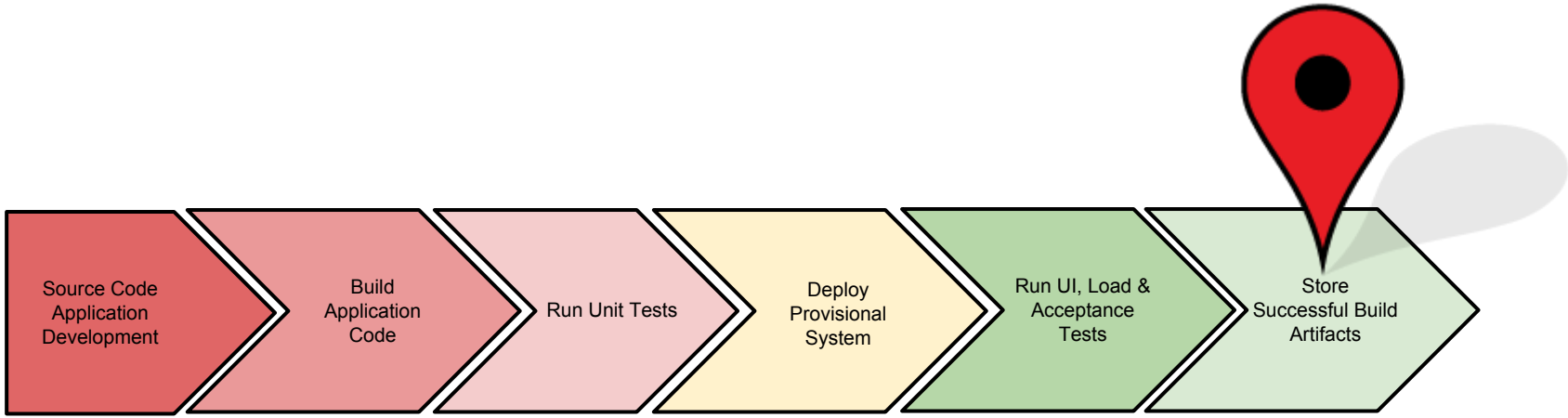
AppVeyor - Test Run

Demo

Break Stuff



OK, We're Ready To Deploy



Azure Websites

DEMO

…But my project isn’t a website

THAT’S OKAY!

Further Considerations

- Provision-As-Needed
- Database Transformations
- Issue Tracking Integration
- Monitoring Metrics Integration (New Relic)
- Machine Image Creation (Atlas)

Wrapping it all up

- Focus on your process
- Define it, discuss it
- Implement & Integrate it
- Iterate and Improve

Questions?