

# Code & Release Management

Eli White

*Zend*

# What are we talking about?

Managing the daily workflow,  
from editing code, to testing and releasing it.

# Version Control

Use it!

It is the core component of this process.

Many variations: All have same core concepts:  
Checkout, Committing, Merging, Concurrency

# Version Control Terminology

## ~ Commit / Check-in

Changes to the code base are added to the repository

## ~ Branch

Making a copy of the code to be managed in parallel

## ~ Tag

Marking a snapshot in time of a set of files

## ~ Trunk

The main line of development, before branching

## ~ Merge

Combining two sets of changes into one

# Subversion (SVN) 101

Subversion thinks in terms of a directory structure

~ Projects are subdirectories of a repository:

```
//host/project
```

~ The mainline (trunk) of code:

```
//host/project/trunk
```

~ Branches & Tags have parallel directories:

```
//host/project/branches/v3.0
```

```
//host/project/tags/v3.0.1
```

# Version Control Policies

Come up with general rules that you apply:

- ~ Intermediate (non-working) checkins?
- ~ Where should you check code in?
- ~ Are there places that are less controlled?
- ~ How does this flow into releases?
- ~ What about tags vs branches vs trunk?

# Uses of Tags/Branches/Trunk

No matter what style of management:

Trunk:

~ Contains the 'core' codebase

Branches:

~ Used to 'segment' into logical areas of responsibility

Tags:

~ Marking a specific state of code, a release

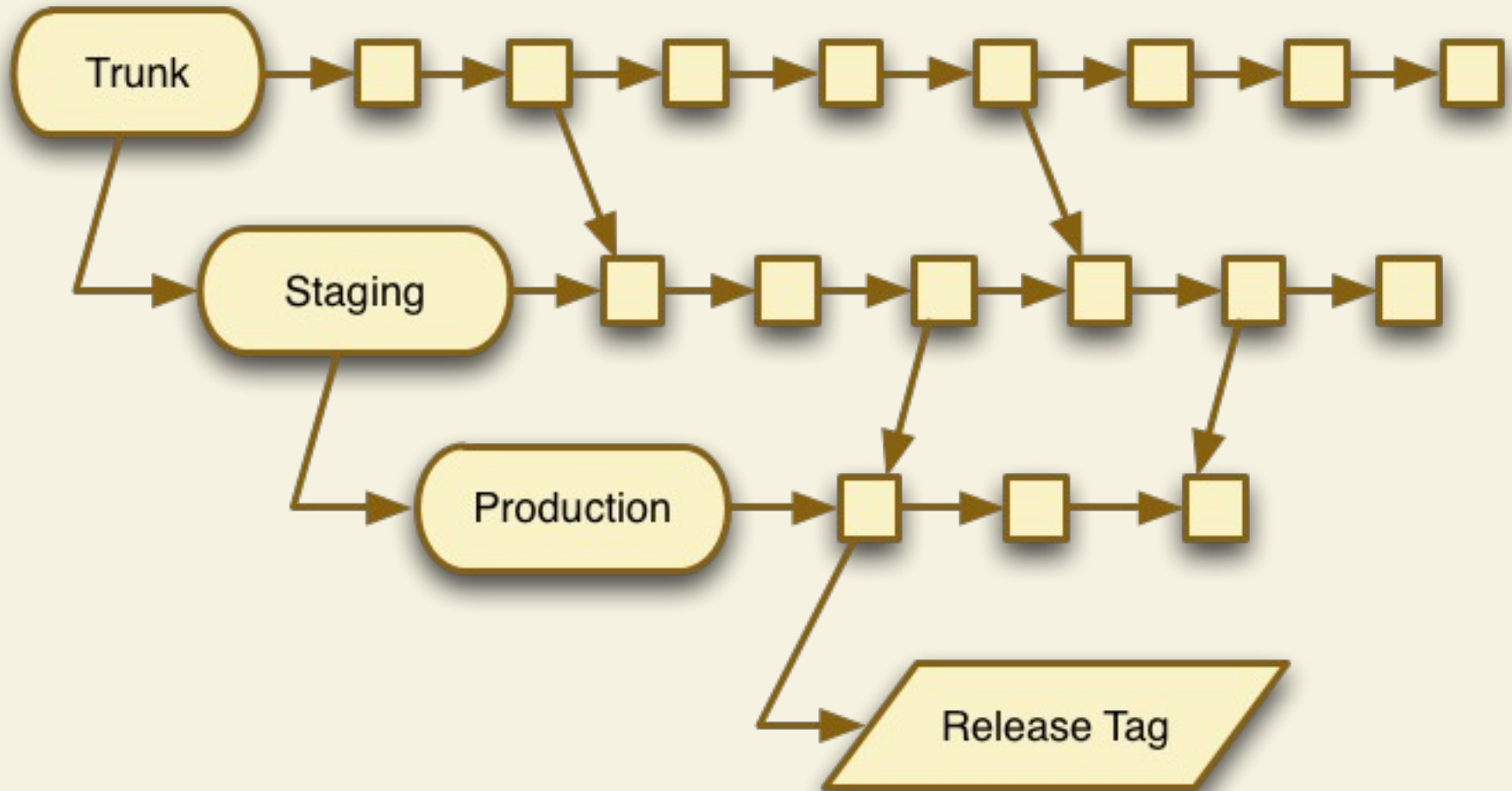
# Explore Three Styles

- ~ Stage Branches
- ~ Feature Branches
- ~ Release Branches

# Stage Branches

- ~ All new work done against Trunk
- ~ Branches exist for each stage of a project:
  - ~ staging, production, etc.
- ~ When ready for a release, merge into staging
- ~ After testing, merge into production
- ~ Tag against production branch for releases

# Stage Branches



# Stage Branches

## Pros:

- ~ Simple
- ~ No dynamic branches

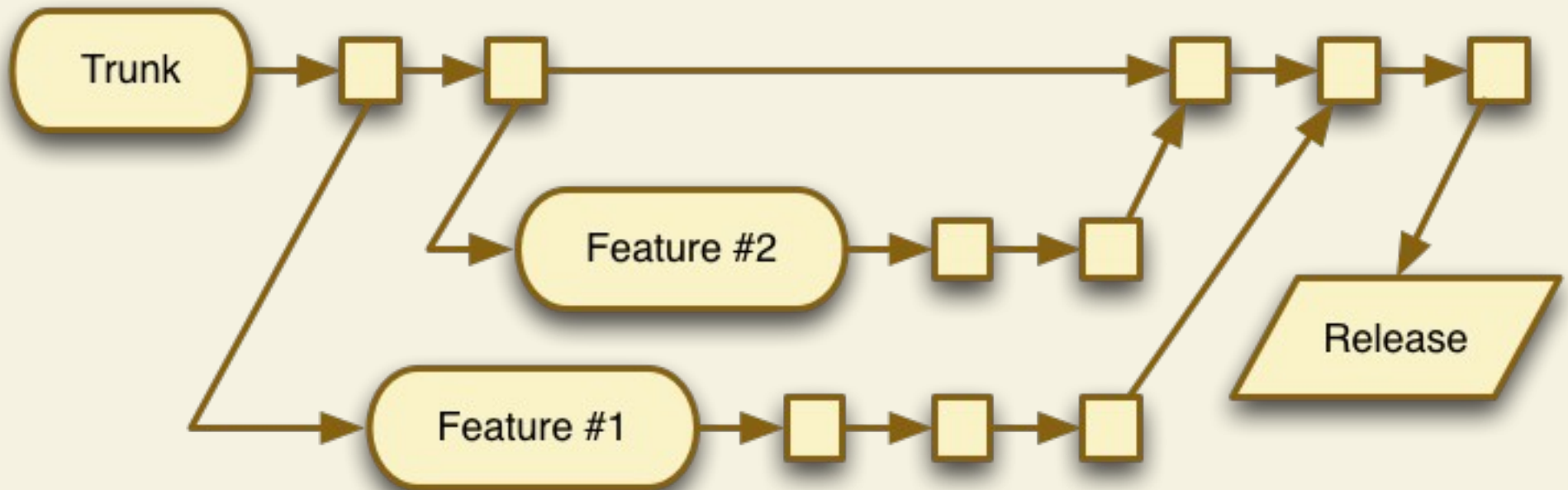
## Cons:

- ~ No parallel work
- ~ No old patches
- ~ No room for errors
- ~ Long scale work hard
- ~ Error prone merging

# Feature Branches

- ~ All new work done in it's own branch
- ~ When complete, the new feature is tested
- ~ Once ready, it's merged to trunk
- ~ Trunk is tagged as needed for phases:
  - ~ For testing/QA, Releasing, etc

# Feature Branches



# Feature Branches

## Pros:

- ~ Parallel work easy
- ~ Long scale work easy

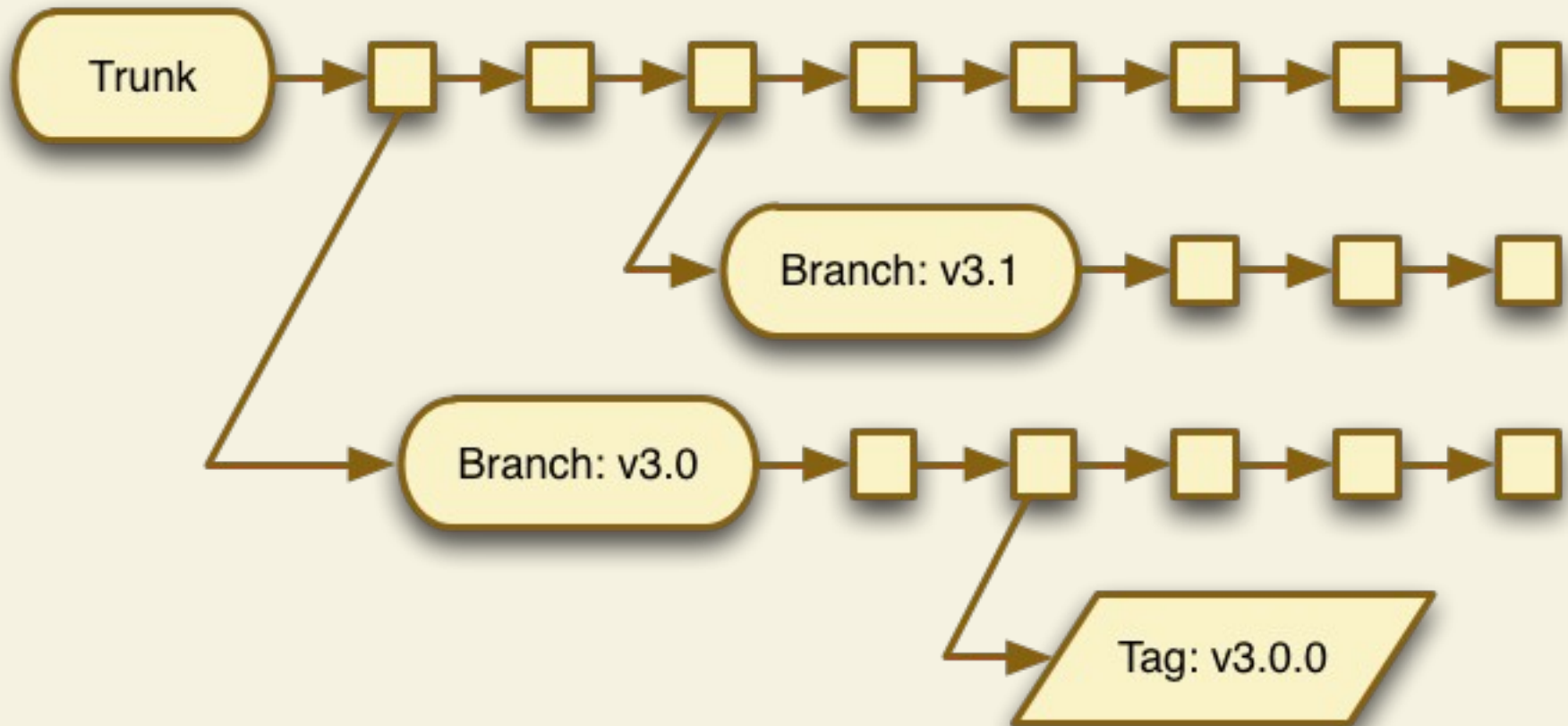
## Cons:

- ~ Often creating branches
- ~ Lots of merging
- ~ No old patches
- ~ Fixes are complicated

# Release Branches

- ~ All new work done on trunk
- ~ When ready for release, create a versional branch: `/branches/v3.0`
- ~ Test against the branch
- ~ Tag the branch with a versional tag for release: `/tags/v3.0.0`
- ~ Fix bugs against branch, and tag as needed
- ~ Continue doing new work against trunk

# Release Branches



# Release Branches

## Pros:

- ~ Easy maintenance
- ~ Long scale work OK
- ~ Some parallel work
- ~ Little merging

## Cons:

- ~ Branch/Tag creation
- ~ Assumes single goal

# Mix and Match

You got feature branches in my release branch!

# Pushing Code Live

Lot's of options, but always based on Tags

~SVN Export & rsync

~Live SVN checkout & update

~Symlink Use

# Questions?

Personal:

<http://eliw.com/>

Work:

<http://devzone.zend.com/>