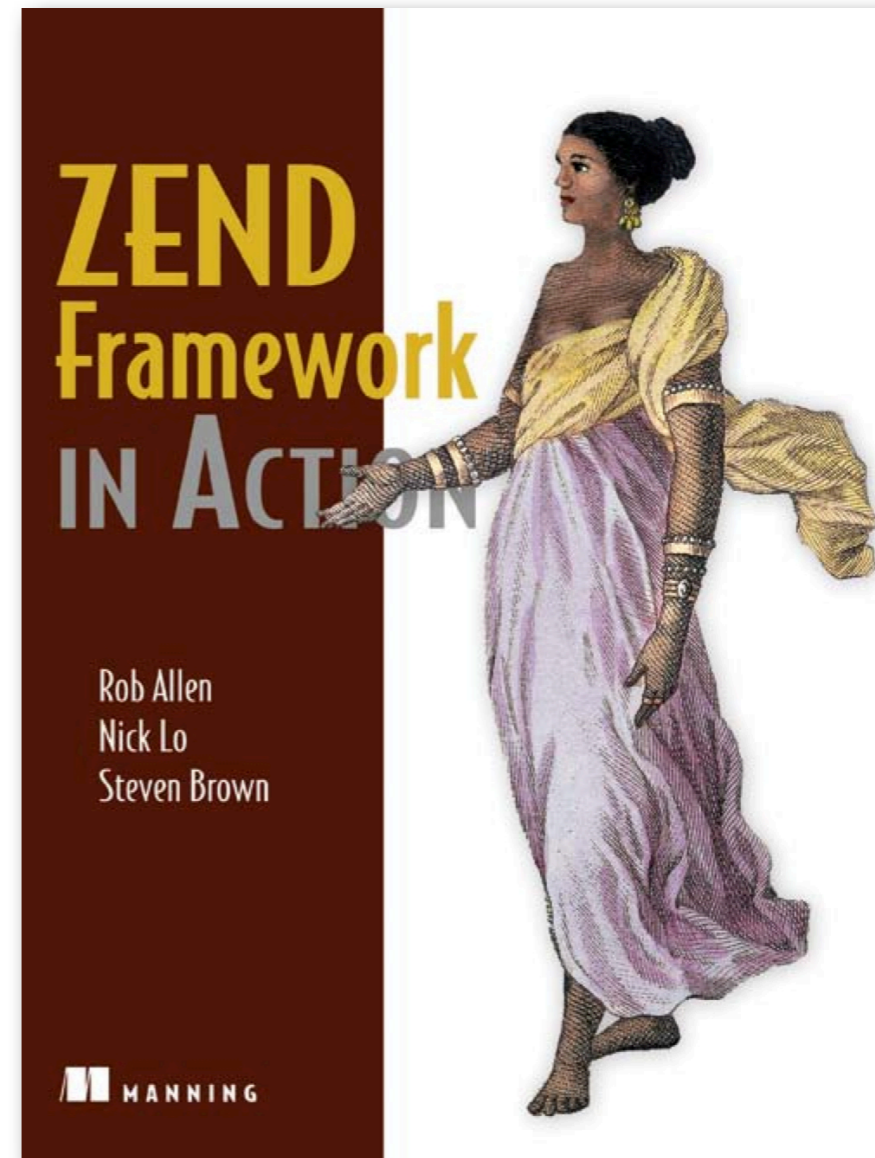


Stress-free Deployment

Rob Allen

Rob Allen?



Why automate deployment?

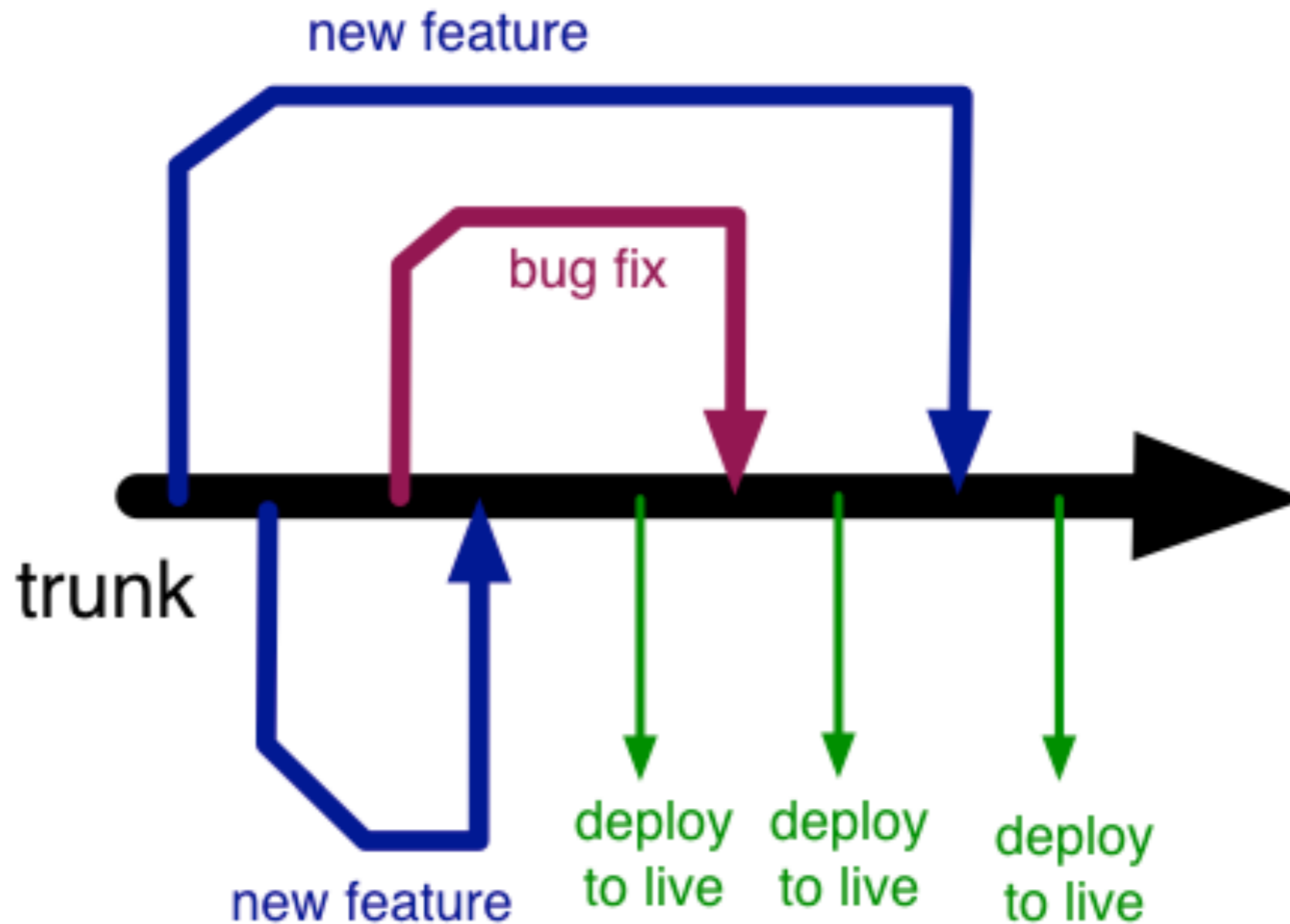
Getting your house in order

Source code control

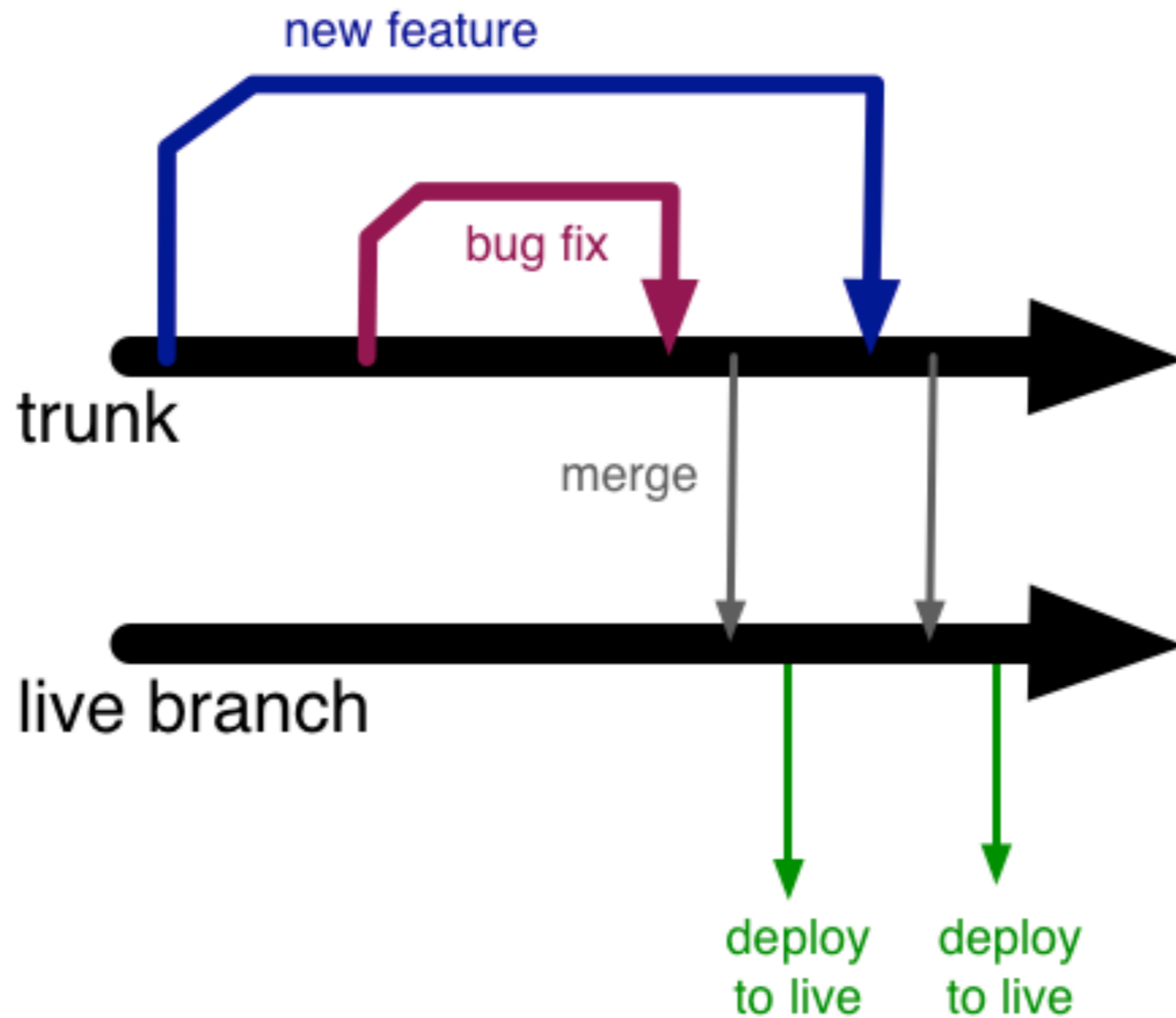
Branch!

- Branch every new feature
 - (that includes bug fixes)
- Be ready go live at all times
 - Trunk deployment
 - Live branch deployment

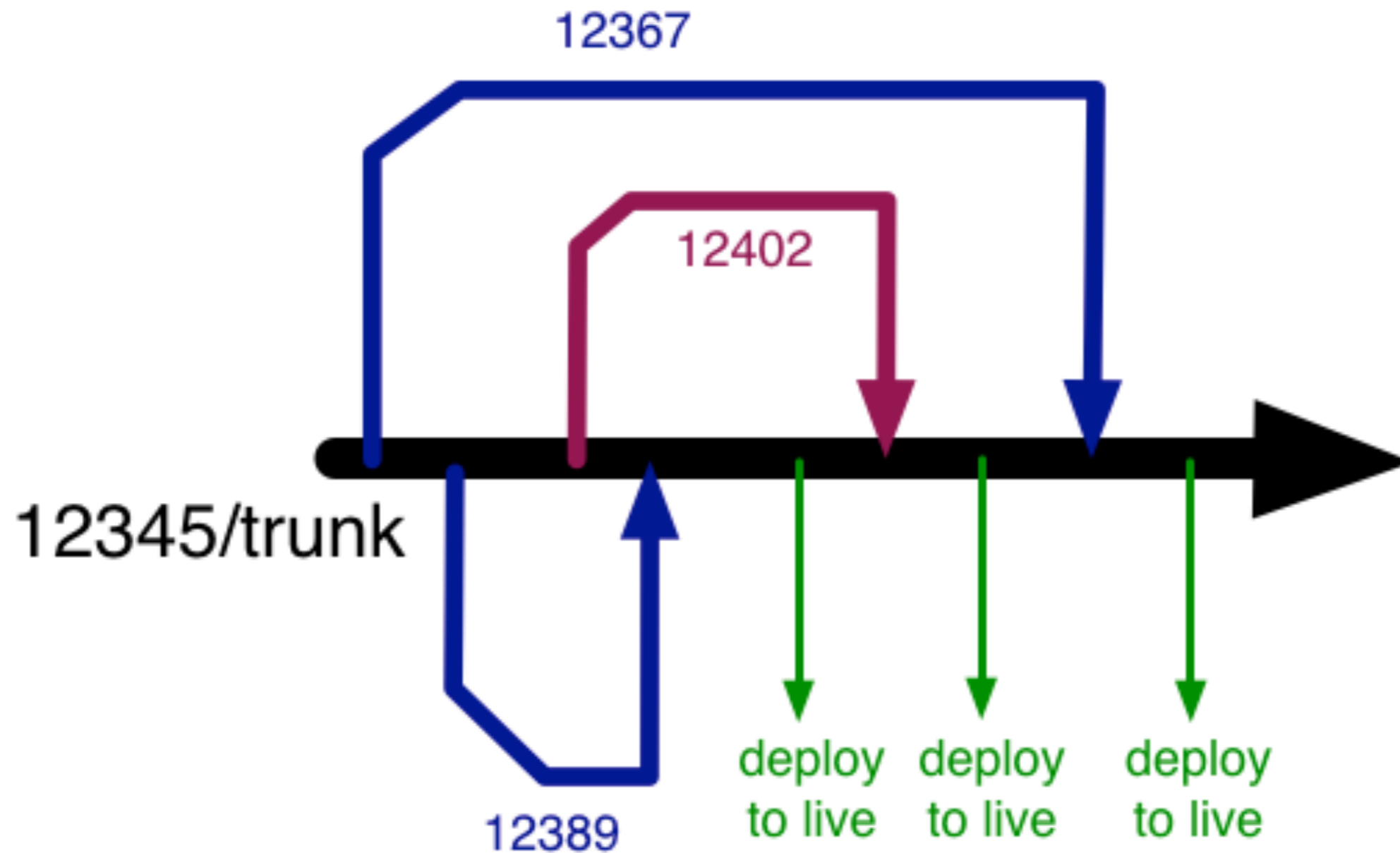
Trunk deployment



Live branch deployment



What do I do?



Database considerations

One master database

- Live database holds master structure
- Copy master to everywhere else
- Advantages:
 - Simple to implement
- Disadvantages:
 - Backwards compatibility required
 - Doesn't scale for multiple devs well
 - Easy to make mistakes

Migrations

- Versioned schemas
- Use *delta* files with UP and DOWN functions
- Advantages:
 - version controlled
 - destructive changes possible
- Disadvantages:
 - Can't think of any!
- Tools:
 - DbDeploy, LiquiBase, Doctrine, homebrew script

Code considerations

Context awareness

- Configuration based on where the code has been deployed
- Automatic detection based on URL?
- Environment variable set in vhost definition?

So what's deployment
all about?

Things to think about

- Transport to server
 - FTP? rsync? svn checkout? svn export?
- File permissions
- Preserve user uploaded files
- Steps after upload
 - Stale cache?
 - Cache priming?

Server organisation

- Much easier if you control vhosts
- For multiple sites on same server:
 - You want predictable file locations for all sites
 - e.g:
 - `/home/www/{site name}/live/current`
 - `/home/www/{site name}/staging/current`

The deployment plan

Typical steps

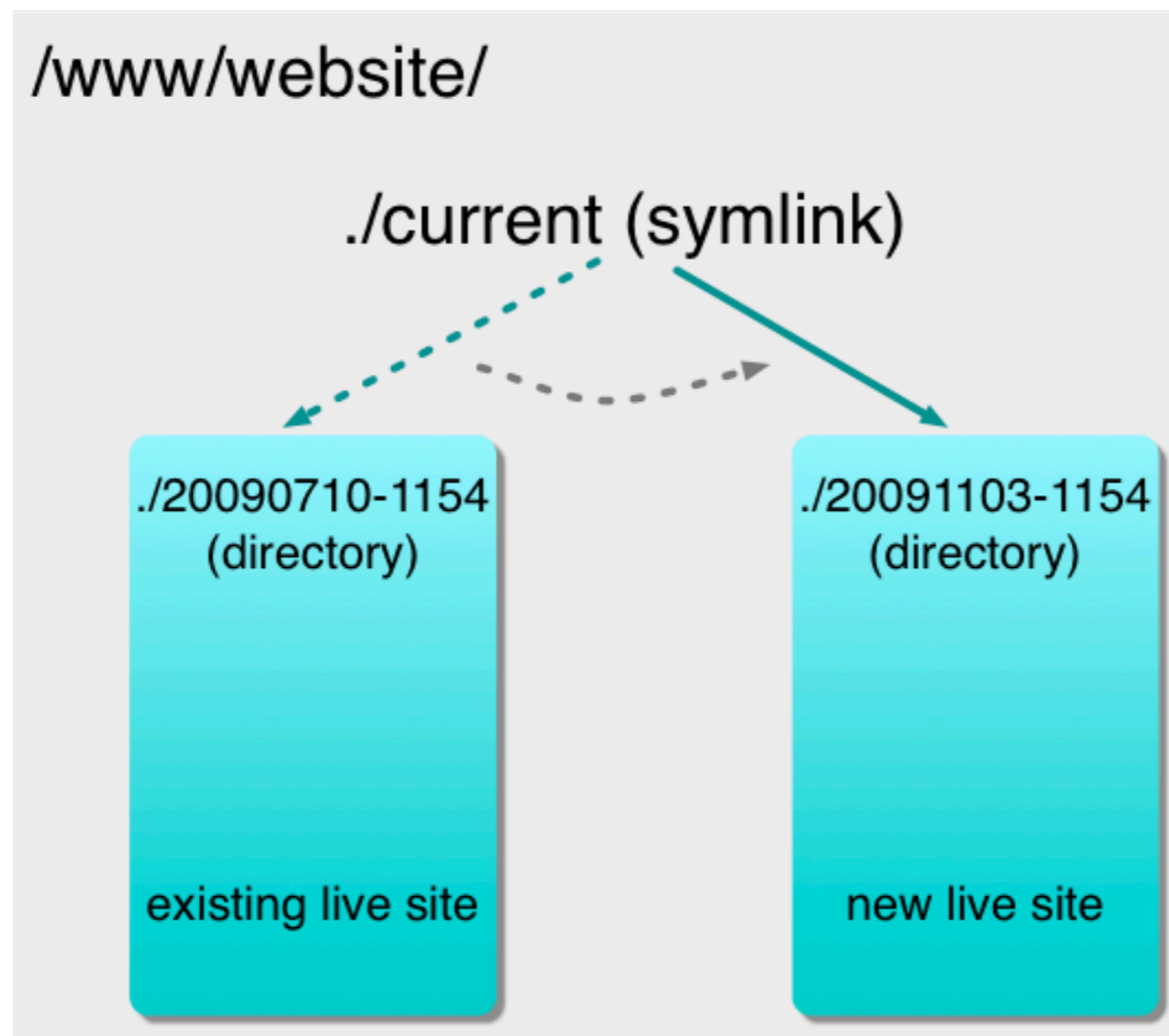
- Tag this release
- Set “under maintenance” page
- Transfer files to server
- Set file permissions as required
- Delete old cache files
- Run database migrations if required
- Remove “under maintenance” page

Here's mine:

- Branch trunk to release-`{yymmdd-hhmm}`
- ssh into server
- Ensure staging is up to date (`svn st -u`)
 - If not, stop here!
- `svn checkout` new release branch to a new folder in live directory
- Set permissions on the `/tmp` folder for cache files

Deployment plan cont.

- Switch “current” symlink to new directory



Tools for automation

Simple scripts

- PHP, Bash or Bat files
- Simple to write and run
- Generally easier to run on the correct server
- Execute command line apps via `exec()`

Example script

```
$cmd = "svn cp -m \"Tag for automatic deployment\"  
      $baseUrl/$website/trunk $baseUrl/$website/tags/$date";  
  
ob_start();  
system($cmd, $returnValue);  
$output = ob_get_clean();  
  
if (0 < $returnValue) {  
    throw new Exception("Tagging failed.\n" . $output);  
}  
echo "Tagged to $date\n";
```


Phing

- PHP based build system based on Ant
- XML configuration files
- PEAR installation
- Integration with Subversion and DbDeploy
- Expects to run `build.xml` in current directory
- `build.properties` contains config info

Phing philosophy

- Like make, build scripts consist of targets
- Targets can depend on other targets
 - “live” depends on “tag”, “checkout”, “migrate”
- Each target does the minimum it can
 - e.g.
 - Create svn tag
 - checkout files to destination
 - migrate database

Example build.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<project name="BRIBuild" default="deploy" basedir=".">
  <tstamp>
    <format property="date" pattern="%Y%m%d-%H%M" />
  </tstamp>
  <property file="build.properties" />
  <property name="trunkpath" value="${svnpath}/${website}/trunk" />
  <property name="tagpath" value="${svnpath}/${website}/tags/${date}" />

  <target name="deploy" depends="tag" />

  <target name="tag" description="Tag trunk">
    <exec command="svn cp -m 'Tag for automatic deployment'
      ${trunkpath} ${tagpath}" />
    <echo msg="Tagged trunk to ${date}" />
  </target>
</project>
```

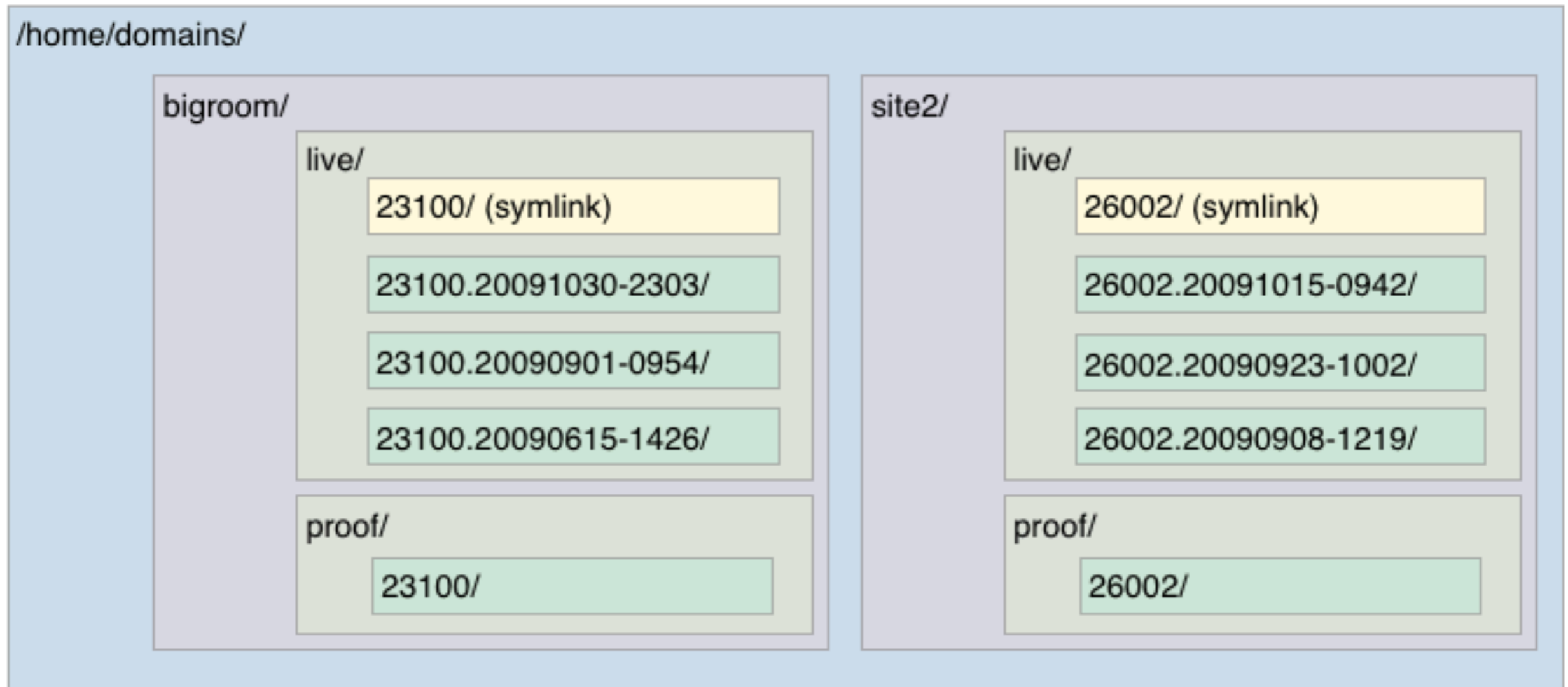
My deployment system

deploy.php

```
~$ deploy.php 23100  
BRI server side deploy script  
Version 1.1, 2009
```

```
Found /home/domains/bigroom/live/  
Tagging to 20091030-2303... done  
Deploying 20091030-2303 ... done  
Changing symlink to new checkout  
Cleaning up older checkouts  
23100 successfully deployed to /home/domains/bigroom/live/
```

Our server layout



deploy.php

- Custom PHP script
- Relies on environment variable: `WWW_DIR`
- Advantages:
 - Custom designed to fit our way of working
 - PHP! Quick and easy to write.
- Disadvantages:
 - Hard to alter for a specific server
 - Hard to change methodology

FTP deployments

```
<project name="project" basedir="." default="deploy">
  <property file="build.properties" />
  <property name="trunkpath" value="${svnpath}/${website}/trunk" />
  <fileset dir="${exportdir}/" id="files">
    <include name="**/*" />
  </fileset>

  <target name="deploy" depends="svnexport,ftp-upload" />

  <target name="svnexport">
    <delete dir="${exportdir}" />
    <svnexport
      username="${username}" password="${password}"
      nocache="true" force="true"
      repositoryurl="${trunkpath}" todir="${exportdir}" />
  </target>
```


FTP deployments

```
<target name="ftp-upload">
  <echo msg="Deploying application files" />
  <ftpdeploy
    host="${ftp.host}" port="${ftp.port}"
    username="${ftp.username}" password="${ftp.password}"
    dir="${ftp.dir}">
    <fileset refid="${files}" />
  </ftpdeploy>
</target>

</project>
```

FTP with Phing

- Per-website build.xml for custom deployments
- Advantages:
 - Leverages other people's experiences
 - Was very fast to create
 - Integrates with xinc
- Disadvantages:
 - New technology to be learnt
 - Phing beta and Pear_Version_SVN alpha

Rollback

Emergency roll-back

Just change the symlink!

Complete roll-back

- Write rollback.php or create a Phing build task
- Put the server back to where it was before
 - Change the symlink
 - Delete the deployed directory
- Database rollback
 - Run down() delta in your migration tool

To summarise

1. Automated deployment prevents mistakes
2. It's not hard
3. Easy roll-back is priceless

Thank you

Please provide feedback on this talk: <http://joind.in/1039>