DRUPALCON LONDON

dog: A New Era for Drupal Sitebuilding

http://drupal.org/project/dog

Sam Boyer (sdboyer)
@sdboyer | gplus.to/sdboyer

Party.
Bus.
After this.
Do it.

At 4PM, buses will start leaving from the main entrance of Fairfield Hall for the official DrupalCon party.

You are attending a vaporware session.

 * Apology - life intervened
 * Reason - this is complicated. needs solid core.

BOF:
 * 11AM tomorrow, room 331 part 1
 * Just curious are welcome
 * NOT just a repeat
 * I'd love contributors. Specific chunks available
 * assume unless i say otherwise that i'm talking about a spec for behavior, not current behavior

dog: “Drupal on Git”

Lots of goals, and will be able to be used to do a bunch of different things. Details later, but I want to quickly give a sense of scope.

<transition>

All of these things are true about dog. But at the end of the day, it’s really about *simplifiying* interaction with your Drupal system as a whole. So instead of having to say, “did I grab the latest db snapshot? Latest files? Cleared caches?” you just give a simple dog command, and it ensures you’re in the right spot. From which comes the moniker I have on the project page – dog is about making Drupal behave simply and well. So you’ll have the desire to say...

<transition>

one note on capitalization, before we get started

dog: “Drupal on Git”

A build system for Drupal, akin to Drush Make

Lots of goals, and will be able to be used to do a bunch of different things. Details later, but I want to quickly give a sense of scope.

<transition>

All of these things are true about dog. But at the end of the day, it’s really about *simplifiying* interaction with your Drupal system as a whole. So instead of having to say, “did I grab the latest db snapshot? Latest files? Cleared caches?” you just give a simple dog command, and it ensures you’re in the right spot. From which comes the moniker I have on the project page – dog is about making Drupal behave simply and well. So you’ll have the desire to say...

<transition>

one note on capitalization, before we get started

dog: “Drupal on Git”

A build system for Drupal, akin to Drush Make

A component of strategies for deployment, configuration management, and continuous integration

Lots of goals, and will be able to be used to do a bunch of different things. Details later, but I want to quickly give a sense of scope.

<transition>

All of these things are true about dog. But at the end of the day, it’s really about *simplifiying* interaction with your Drupal system as a whole. So instead of having to say, “did I grab the latest db snapshot? Latest files? Cleared caches?” you just give a simple dog command, and it ensures you’re in the right spot. From which comes the moniker I have on the project page - dog is about making Drupal behave simply and well. So you’ll have the desire to say...

<transition>

one note on capitalization, before we get started

dog: “Drupal on Git”

A build system for Drupal, akin to Drush Make

A component of strategies for deployment, configuration management, and continuous integration

A system to ease working with upstream Drupal code - both contributing patches and managing updates

Lots of goals, and will be able to be used to do a bunch of different things. Details later, but I want to quickly give a sense of scope.

<transition>

All of these things are true about dog. But at the end of the day, it’s really about *simplifiying* interaction with your Drupal system as a whole. So instead of having to say, “did I grab the latest db snapshot? Latest files? Cleared caches?” you just give a simple dog command, and it ensures you’re in the right spot. From which comes the moniker I have on the project page – dog is about making Drupal behave simply and well. So you’ll have the desire to say...

<transition>

one note on capitalization, before we get started

dog: “Drupal on Git”

A build system for Drupal, akin to Drush Make

A component of strategies for deployment, configuration management, and continuous integration

A system to ease working with upstream Drupal code - both contributing patches and managing updates

A system for capturing and managing a Drupal site’s overall state

Lots of goals, and will be able to be used to do a bunch of different things. Details later, but I want to quickly give a sense of scope.

<transition>

All of these things are true about dog. But at the end of the day, it’s really about *simplifiying* interaction with your Drupal system as a whole. So instead of having to say, “did I grab the latest db snapshot? Latest files? Cleared caches?” you just give a simple dog command, and it ensures you’re in the right spot. From which comes the moniker I have on the project page - dog is about making Drupal behave simply and well. So you’ll have the desire to say...

<transition>

one note on capitalization, before we get started

dog: “Drupal on Git”

A build system for Drupal, akin to Drush Make

A component of strategies for deployment, configuration management, and continuous integration

A system to ease working with upstream Drupal code - both contributing patches and managing updates

A system for capturing and managing a Drupal site’s overall state

Sit Drupal, sit. Good dog.

Lots of goals, and will be able to be used to do a bunch of different things. Details later, but I want to quickly give a sense of scope.

<transition>

All of these things are true about dog. But at the end of the day, it’s really about *simplifiying* interaction with your Drupal system as a whole. So instead of having to say, “did I grab the latest db snapshot? Latest files? Cleared caches?” you just give a simple dog command, and it ensures you’re in the right spot. From which comes the moniker I have on the project page - dog is about making Drupal behave simply and well. So you’ll have the desire to say...

<transition>

one note on capitalization, before we get started

This is a DOG...

one last quick note on capitalization. i have a weird obsession with lowercase in general, but you'll notice that i'm extra-adamant about not capitalizing dog at all, despite it clearly being an acronym.

Now keep in mind that I'm an animal lover in general and a dog lover in particular. Well-trained military dogs aren't necessarily feral or cruel by nature - they have a job they've been trained to do, but they're just as loyal and capable of love as any other dog.

But given the choice for the kinda work we need done here, I’m gonna take the nerdy little hipster pooch over giant jock attack dog every time. And since that’s what the different spellings conjure up for me...

And yes, it is the offiical policy of the project to encourage dog puns. The groanier the better. This may be the only non-joke project on drupal.org that will accept patches based purely on pun quality.

Site management woes

So you’re supposed to start a presentation like this with a statement of the problem you’re solving. Yeah, I tried, but I’m bad at that. I never know the right order to present stuff in. So here’s my attempt. But when I’m done, I’m just gonna tell a story.
<<
 * We’ve started to see a portable format with Pantheon and Dev Cloud settling on the site tarball. This is absolutely the same idea
<<
 * We often say code “lives” in a repo. Reflect on that. Taking it out of that repo (in a tarball) *kills* the code, because it separates it from its history
 * Patching without history sucks. It’s like flying blind.
 *

Story time! This is how *I* came around to the idea of dog.

Site management woes

- Drupal sites, in their entirety, aren’t especially portable

So you’re supposed to start a presentation like this with a statement of the problem you’re solving. Yeah, I tried, but I’m bad at that. I never know the right order to present stuff in. So here’s my attempt. But when I’m done, I’m just gonna tell a story.
<<
 * We’ve started to see a portable format with Pantheon and Dev Cloud settling on the site tarball. This is absolutely the same idea
<<
 * We often say code “lives” in a repo. Reflect on that. Taking it out of that repo (in a tarball) *kills* the code, because it separates it from its history
 * Patching without history sucks. It’s like flying blind.
 *

Story time! This is how *I* came around to the idea of dog.

Site management woes

- Drupal sites, in their entirety, aren’t especially portable
    - Code + database + files + ??

So you’re supposed to start a presentation like this with a statement of the problem you’re solving. Yeah, I tried, but I’m bad at that. I never know the right order to present stuff in. So here’s my attempt. But when I’m done, I’m just gonna tell a story.
<<
 * We’ve started to see a portable format with Pantheon and Dev Cloud settling on the site tarball. This is absolutely the same idea
<<
 * We often say code “lives” in a repo. Reflect on that. Taking it out of that repo (in a tarball) *kills* the code, because it separates it from its history
 * Patching without history sucks. It’s like flying blind.
 *

Story time! This is how *I* came around to the idea of dog.

Site management woes

- Drupal sites, in their entirety, aren't especially portable
    - Code + database + files + ??
    - Makes it (prohibitively) difficult to work across multiple instances

So you're supposed to start a presentation like this with a statement of the problem you're solving. Yeah, I tried, but I'm bad at that. I never know the right order to present stuff in. So here's my attempt. But when I'm done, I'm just gonna tell a story.
<<
 * We've started to see a portable format with Pantheon and Dev Cloud settling on the site tarball. This is absolutely the same idea
<<
 * We often say code "lives" in a repo. Reflect on that. Taking it out of that repo (in a tarball) *kills* the code, because it separates it from its history
 * Patching without history sucks. It's like flying blind.
 *

Story time! This is how *I* came around to the idea of dog.

Site management woes

- Drupal sites, in their entirety, aren't especially portable
  - Code + database + files + ??
  - Makes it (prohibitively) difficult to work across multiple instances
- Code from tarballs is dead code

So you're supposed to start a presentation like this with a statement of the problem you're solving. Yeah, I tried, but I'm bad at that. I never know the right order to present stuff in. So here's my attempt. But when I'm done, I'm just gonna tell a story.
<<
 * We've started to see a portable format with Pantheon and Dev Cloud settling on the site tarball. This is absolutely the same idea
<<
 * We often say code "lives" in a repo. Reflect on that. Taking it out of that repo (in a tarball) *kills* the code, because it separates it from its history
 * Patching without history sucks. It's like flying blind.
 *

Story time! This is how *I* came around to the idea of dog.

Site management woes

- Drupal sites, in their entirety, aren't especially portable
  - Code + database + files + ??
  - Makes it (prohibitively) difficult to work across multiple instances
- Code from tarballs is dead code
  - Patching modules for a real site sucks

So you're supposed to start a presentation like this with a statement of the problem you're solving. Yeah, I tried, but I'm bad at that. I never know the right order to present stuff in. So here's my attempt. But when I'm done, I'm just gonna tell a story.
<<
 * We've started to see a portable format with Pantheon and Dev Cloud settling on the site tarball. This is absolutely the same idea
<<
 * We often say code "lives" in a repo. Reflect on that. Taking it out of that repo (in a tarball) *kills* the code, because it separates it from its history
 * Patching without history sucks. It's like flying blind.
 *

Story time! This is how *I* came around to the idea of dog.

Site management woes

- Drupal sites, in their entirety, aren’t especially portable
  - Code + database + files + ??
  - Makes it (prohibitively) difficult to work across multiple instances
- Code from tarballs is dead code
  - Patching modules for a real site sucks
- *Drush (Make) helps with **some** of this*

So you’re supposed to start a presentation like this with a statement of the problem you’re solving. Yeah, I tried, but I’m bad at that. I never know the right order to present stuff in. So here’s my attempt. But when I’m done, I’m just gonna tell a story.
<<
 * We’ve started to see a portable format with Pantheon and Dev Cloud settling on the site tarball. This is absolutely the same idea
<<
 * We often say code “lives” in a repo. Reflect on that. Taking it out of that repo (in a tarball) *kills* the code, because it separates it from its history
 * Patching without history sucks. It’s like flying blind.
 *

Story time! This is how *I* came around to the idea of dog.

Mr. Module, can we be friends?

On first consideration of a module for use on a site, I:

 * read project page
 * read README, etc.
 * i may also install it on a dev site and quickly poke through the UI, but at some point i will ALWAYS...

<transition>

 * open it up and look through the code
 * special time with the module. strong chance i will NEVER look at it so closely again

Mr. Module, can we be friends?

```
<?php

/**
 * @file might_be_good.module
 *
 * I'm tantalizing. My project description looks
 * spot-on for what you need. But can you REALLY
 * trust me? Only my code will speak the
 * unvarnished truth...
 */
```

On first consideration of a module for use on a site, I:

 * read project page
 * read README, etc.
 * i may also install it on a dev site and quickly poke through the UI, but at some point i will ALWAYS...

<transition>

 * open it up and look through the code
 * special time with the module. strong chance i will NEVER look at it so closely again

```
/**
 * Implements hook_node_load().
 *
 */
function might_be_good_node_load(&$nodes, $types) {
```

 * Notice function signature is not quite right
 * Not a critical bug, though maybe a little worrisome

 * Even more minor, we have a typo in docs

These aren’t show-stopper problems, but I might like to do the module maintainer a solid and throw some quick fixes into the queue.

```
/**
 * Implements hook_node_load().
 *
 */
function might_be_good_node_load(&$nodes, $types) {
```

```
/**
 * Impelments hook_foo().
 *
 */
function might_be_good_foo() {
```

* Notice function signature is not quite right
* Not a critical bug, though maybe a little worrisome

* Even more minor, we have a typo in docs

These aren’t show-stopper problems, but I might like to do the module maintainer a solid and throw some quick fixes into the queue.

“Quick Fix” from a tarball

(narrate through each step)

The process borders on being arduous. At minimum it is *certainly* disruptive enough that I won’t do it unless I’m looking for an excuse to procrastinate.

“Quick Fix” from a tarball

- Get rid of the code I already have and replace it with -dev

(narrate through each step)

The process borders on being arduous. At minimum it is *certainly* disruptive enough that I won’t do it unless I’m looking for an excuse to procrastinate.

“Quick Fix” from a tarball

- Get rid of the code I already have and replace it with -dev
- Check and make sure the problem(s) are still there

(narrate through each step)

The process borders on being arduous. At minimum it is *certainly* disruptive enough that I won’t do it unless I’m looking for an excuse to procrastinate.

“Quick Fix” from a tarball

- Get rid of the code I already have and replace it with -dev
- Check and make sure the problem(s) are still there
- Create a patch file with my fixes

(narrate through each step)

The process borders on being arduous. At minimum it is *certainly* disruptive enough that I won’t do it unless I’m looking for an excuse to procrastinate.

“Quick Fix” from a tarball

- Get rid of the code I already have and replace it with -dev
- Check and make sure the problem(s) are still there
- Create a patch file with my fixes
- Post a new issue with the patch

(narrate through each step)

The process borders on being arduous. At minimum it is *certainly* disruptive enough that I won’t do it unless I’m looking for an excuse to procrastinate.

“Quick Fix” from a tarball

- Get rid of the code I already have and replace it with -dev
- Check and make sure the problem(s) are still there
- Create a patch file with my fixes
- Post a new issue with the patch
- Keep the patchfile locally version controlled so that it can be re-run after future upstream updates blow it away

(narrate through each step)

The process borders on being arduous. At minimum it is *certainly* disruptive enough that I won’t do it unless I’m looking for an excuse to procrastinate.

Quick fix from a Git repo

This is a cakewalk, at least for me. But that's part of what's great here: you're using the same skillset and techniques to work on your local site as you do to contribute to Drupal. In fact, what we've really done here is yank the bottom out from under the "barrier to contributions". I really don't think it's possible to overstate the importance of this for continuing to broaden participation in our community.

So with this cool workflow benefit in mind, the next natural question was – can we scale this up? what does it look like?

Quick fix from a Git repo

- Check against -dev: `git checkout origin/7.x-1.x` and look for the bug

This is a cakewalk, at least for me. But that's part of what's great here: you're using the same skillset and techniques to work on your local site as you do to contribute to Drupal. In fact, what we've really done here is yank the bottom out from under the "barrier to contributions". I really don't think it's possible to overstate the importance of this for continuing to broaden participation in our community.

So with this cool workflow benefit in mind, the next natural question was – can we scale this up? what does it look like?

Quick fix from a Git repo

- Check against -dev: **`git checkout origin/7.x-1.x`** and look for the bug
- Create a local branch from the stable tag: **`git checkout -b master 7.x-1.0`** then directly commit my fix(es)

This is a cakewalk, at least for me. But that's part of what's great here: you're using the same skillset and techniques to work on your local site as you do to contribute to Drupal. In fact, what we've really done here is yank the bottom out from under the "barrier to contributions". I really don't think it's possible to overstate the importance of this for continuing to broaden participation in our community.

So with this cool workflow benefit in mind, the next natural question was – can we scale this up? what does it look like?

Quick fix from a Git repo

- Check against -dev: `git checkout origin/7.x-1.x` and look for the bug
- Create a local branch from the stable tag: `git checkout -b master 7.x-1.0` then directly commit my fix(es)
- Send the code back up to an issue. (Currently this still means patches, but per-issue repositories are on the drupal.org roadmap)

This is a cakewalk, at least for me. But that's part of what's great here: you're using the same skillset and techniques to work on your local site as you do to contribute to Drupal. In fact, what we've really done here is yank the bottom out from under the "barrier to contributions". I really don't think it's possible to overstate the importance of this for continuing to broaden participation in our community.

So with this cool workflow benefit in mind, the next natural question was – can we scale this up? what does it look like?

Quick fix from a Git repo

- Check against -dev: **`git checkout origin/7.x-1.x`** and look for the bug
- Create a local branch from the stable tag: **`git checkout -b master 7.x-1.0`** then directly commit my fix(es)
- Send the code back up to an issue. (Currently this still means patches, but per-issue repositories are on the drupal.org roadmap)
- When upstream updates come in the future, Git's *real* merge mechanics do the job better than re-running **`patch`** ever could

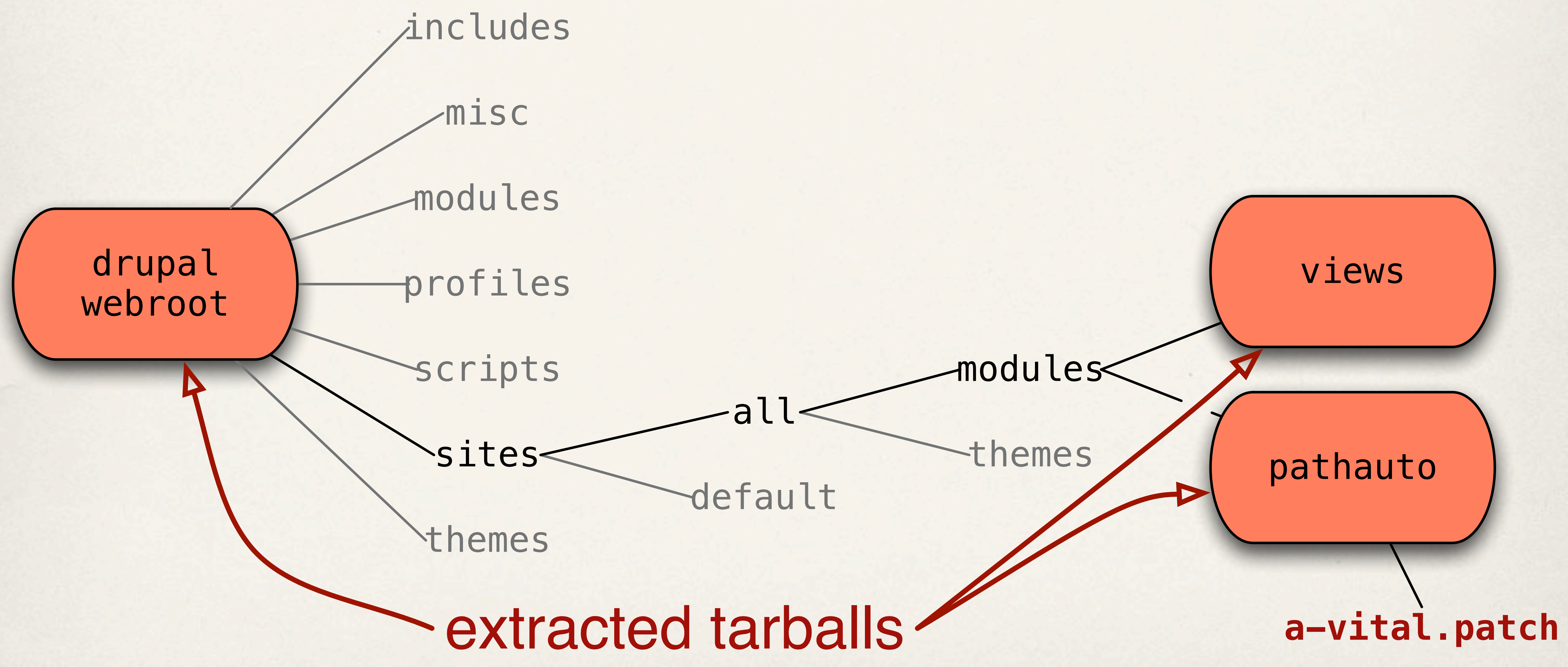
This is a cakewalk, at least for me. But that's part of what's great here: you're using the same skillset and techniques to work on your local site as you do to contribute to Drupal. In fact, what we've really done here is yank the bottom out from under the "barrier to contributions". I really don't think it's possible to overstate the importance of this for continuing to broaden participation in our community.

So with this cool workflow benefit in mind, the next natural question was – can we scale this up? what does it look like?

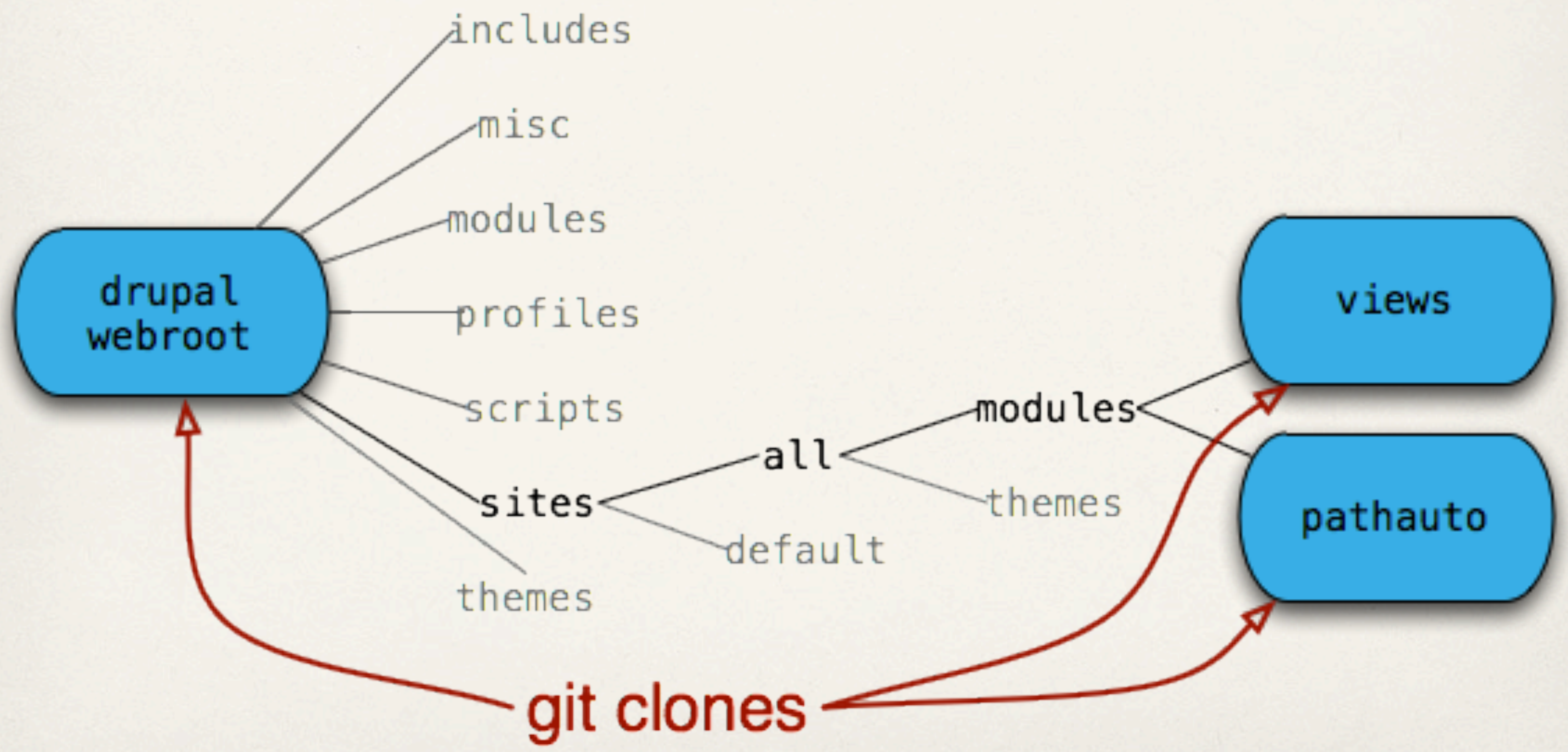
Let’s draw this out a bit more concretely...

So instead of some hypothetical module, let’s imagine a Drupal tree with Views and Pathauto installed, where we need to patch pathauto.

Patching contrib with tarballs



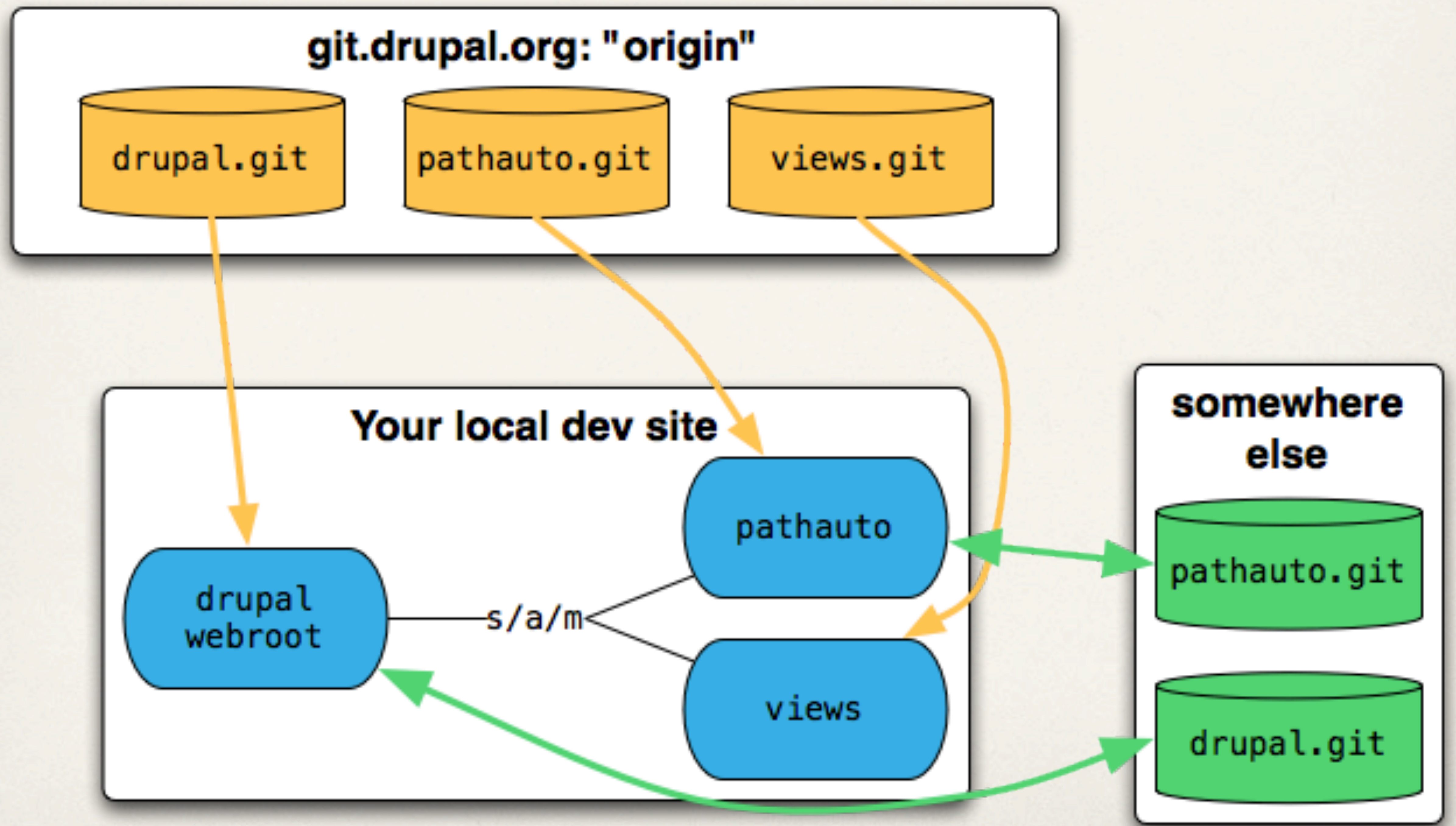
Patching contrib with Git



so here we have the same basic tree as we did with the tarballs, but with git repos + worktrees instead.

those who've tinkered with stuff like this will note that this example does NOT use submodules here. don't infer too much from that, i'll discuss it more later.

Problem already: lotsa remotes!



of course, to make this all work, we're gonna have to juggle a bunch of remotes. we've got the remotes we cloned from on drupal.org, and then for every repository we make a change in, they each need their own repo we can write to.

this was starting to look like a hot mess. for this to be feasible, we really needed a way of managing repo clusters easily.

What will make using clusters of Git repos easy?

 * well, if we can make multiple repos act like one, that's kinda the obvious answer
 * ok so that means network & tree level operations need to be unified, as well a Drush Make-like build actions
 * as a general rule, git-submodule sucks and is very frustrating to use. we need alternatives! only good for libraries that you *never* touch.
 * Local branch naming, local hooks
 * GUI users - don't have plans to touch on this much today, but my hope is that we can effectively utilize git's local hooks to make GUIs still passable

What will make using clusters of Git repos easy?

- Simplicity of a single repo with the power of multiple

 * well, if we can make multiple repos act like one, that's kinda the obvious answer
 * ok so that means network & tree level operations need to be unified, as well a Drush Make-like build actions
 * as a general rule, git-submodule sucks and is very frustrating to use. we need alternatives! only good for libraries that you *never* touch.
 * Local branch naming, local hooks
 * GUI users - don't have plans to touch on this much today, but my hope is that we can effectively utilize git's local hooks to make GUIs still passable

What will make using clusters of Git repos easy?

- Simplicity of a single repo with the power of multiple
    - **`git push`**, **`git pull`**, change branches, and build all together

 * well, if we can make multiple repos act like one, that’s kinda the obvious answer
 * ok so that means network & tree level operations need to be unified, as well a Drush Make-like build actions
 * as a general rule, git-submodule sucks and is very frustrating to use. we need alternatives! only good for libraries that you *never* touch.
 * Local branch naming, local hooks
 * GUI users - don’t have plans to touch on this much today, but my hope is that we can effectively utilize git’s local hooks to make GUIs still passable

What will make using clusters of Git repos easy?

- Simplicity of a single repo with the power of multiple
  - **`git push`**, **`git pull`**, change branches, and build all together
  - Need to sand down the ugliness of Git’s submodules

* well, if we can make multiple repos act like one, that’s kinda the obvious answer
* ok so that means network & tree level operations need to be unified, as well a Drush Make-like build actions
* as a general rule, git-submodule sucks and is very frustrating to use. we need alternatives! only good for libraries that you *never* touch.
* Local branch naming, local hooks
* GUI users - don’t have plans to touch on this much today, but my hope is that we can effectively utilize git’s local hooks to make GUIs still passable

What will make using clusters of Git repos easy?

- Simplicity of a single repo with the power of multiple
    - **`git push`**, **`git pull`**, change branches, and build all together
    - Need to sand down the ugliness of Git's submodules
- Ensure consistent behavior across different local git repos

 * well, if we can make multiple repos act like one, that's kinda the obvious answer
 * ok so that means network & tree level operations need to be unified, as well a Drush Make-like build actions
 * as a general rule, git-submodule sucks and is very frustrating to use. we need alternatives! only good for libraries that you *never* touch.
 * Local branch naming, local hooks
 * GUI users - don't have plans to touch on this much today, but my hope is that we can effectively utilize git's local hooks to make GUIs still passable

What will make using clusters of Git repos easy?

- Simplicity of a single repo with the power of multiple
  - **`git push`, `git pull`**, change branches, and build all together
  - Need to sand down the ugliness of Git's submodules
- Ensure consistent behavior across different local git repos
- And hey, while we're at it, support moving around *more* than just code: capture, record, and replay the overall state of a Drupal site

 * well, if we can make multiple repos act like one, that's kinda the obvious answer
 * ok so that means network & tree level operations need to be unified, as well a Drush Make-like build actions
 * as a general rule, git-submodule sucks and is very frustrating to use. we need alternatives! only good for libraries that you *never* touch.
 * Local branch naming, local hooks
 * GUI users - don't have plans to touch on this much today, but my hope is that we can effectively utilize git's local hooks to make GUIs still passable

What will make using clusters of Git repos easy?

- Simplicity of a single repo with the power of multiple
  - **`git push`, `git pull`**, change branches, and build all together
  - Need to sand down the ugliness of Git's submodules
- Ensure consistent behavior across different local git repos
- And hey, while we're at it, support moving around *more* than just code: capture, record, and replay the overall state of a Drupal site
- Can't forget GUI users!

 * well, if we can make multiple repos act like one, that's kinda the obvious answer
 * ok so that means network & tree level operations need to be unified, as well a Drush Make-like build actions
 * as a general rule, git-submodule sucks and is very frustrating to use. we need alternatives! only good for libraries that you *never* touch.
 * Local branch naming, local hooks
 * GUI users - don't have plans to touch on this much today, but my hope is that we can effectively utilize git's local hooks to make GUIs still passable

And so, the great luchadog pledges his assistance.

Yes, this is basically dog: making working with Drupal, and clusters of Git repos, easier.

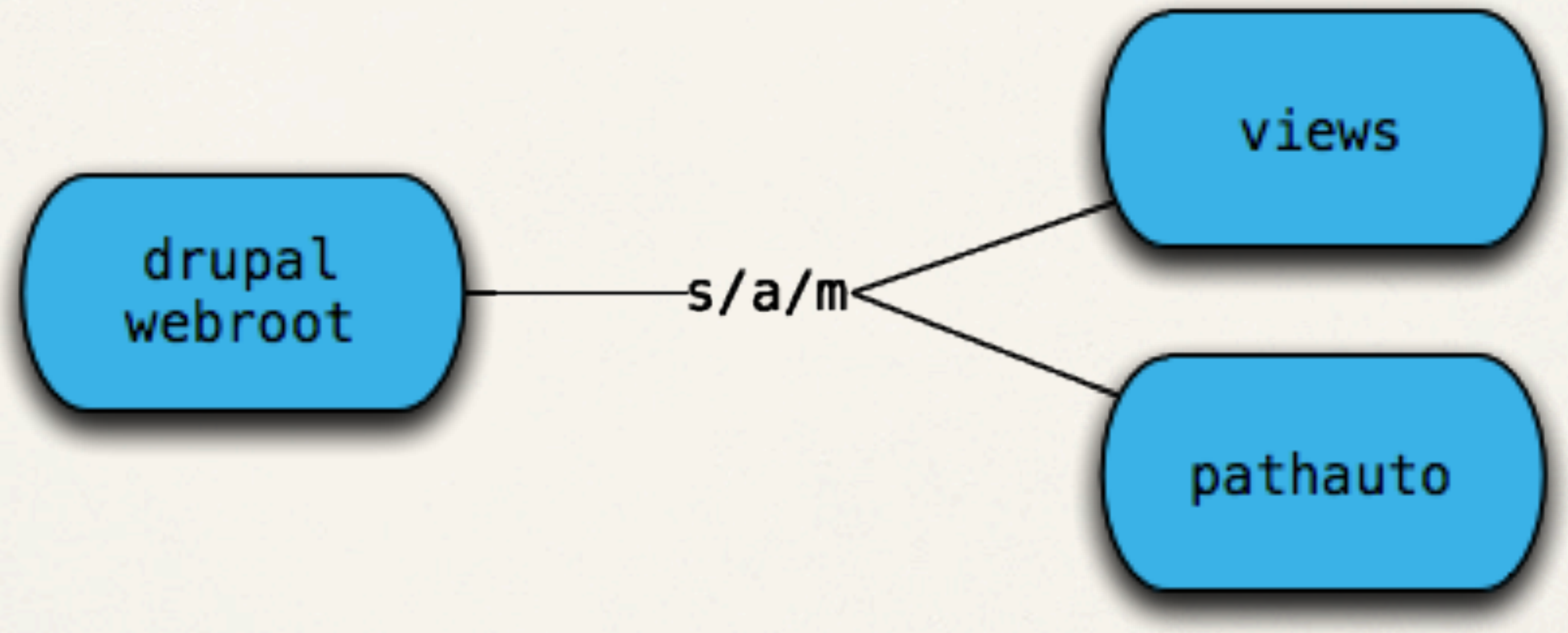
Let’s look at how dog does that.

And so, the great luchadog pledges his assistance.

Yes, this is basically dog: making working with Drupal, and clusters of Git repos, easier.

Let’s look at how dog does that.

Good dogs need houses. And sleds.



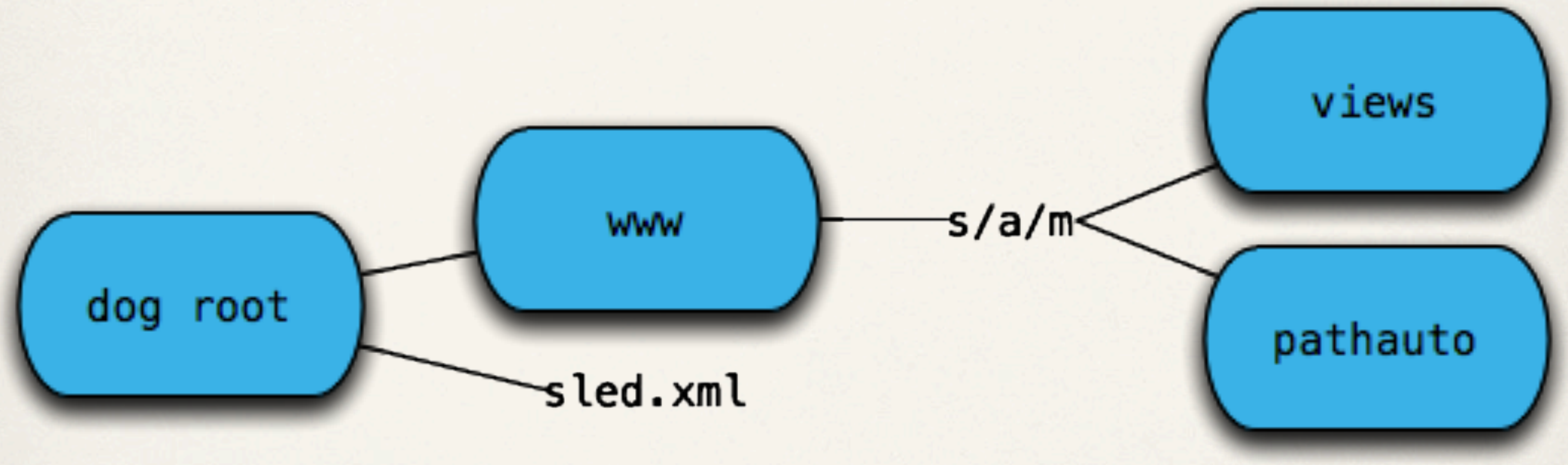
* DOG HOUSE!

so this is what dog systems actually start to look like.

Especially as sites become more complex, it becomes a better and better idea to NOT have the root of your repository correspond to the webroot. There are often scripts (or maybe even tests – PHPUnit, Selenium!!), etc. that you need as part of the project, and they make no sense under the webroot.

So dog kicks out a parent repository to contain this. This repository has no upstream. And it’s called the Dog House! muahahahaha

Good dogs need houses. And sleds.



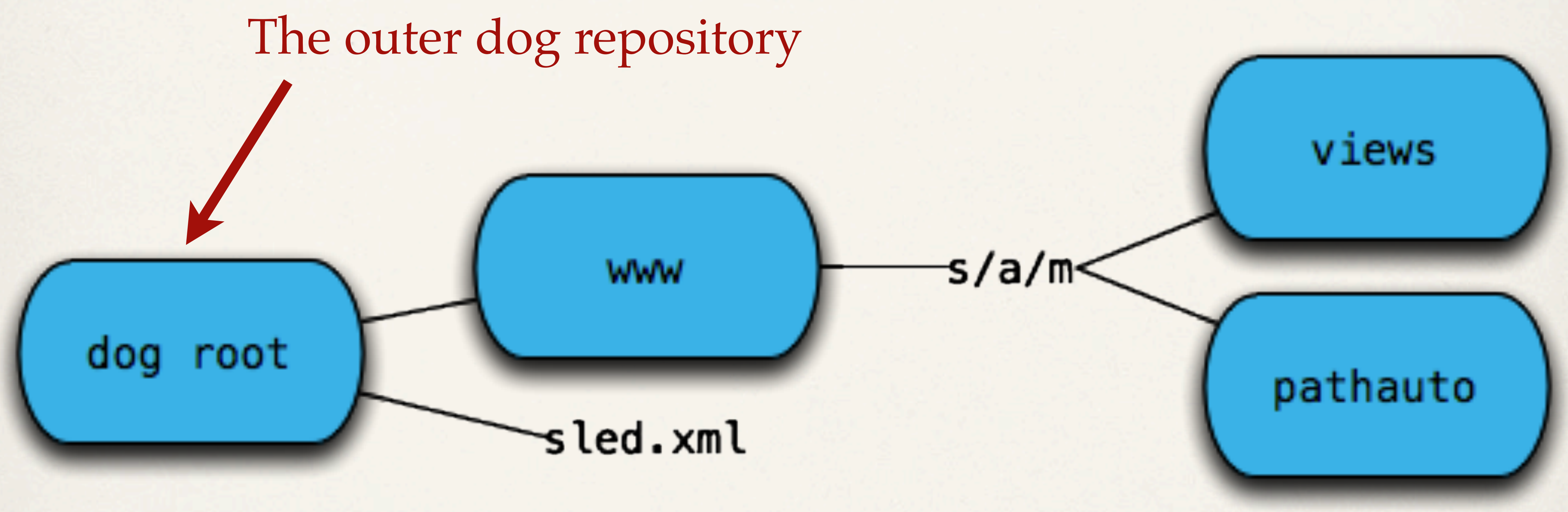
* DOG HOUSE!

so this is what dog systems actually start to look like.

Especially as sites become more complex, it becomes a better and better idea to NOT have the root of your repository correspond to the webroot. There are often scripts (or maybe even tests - PHPUnit, Selenium!!), etc. that you need as part of the project, and they make no sense under the webroot.

So dog kicks out a parent repository to contain this. This repository has no upstream. And it’s called the Dog House! muahahahaha

Good dogs need houses. And sleds.



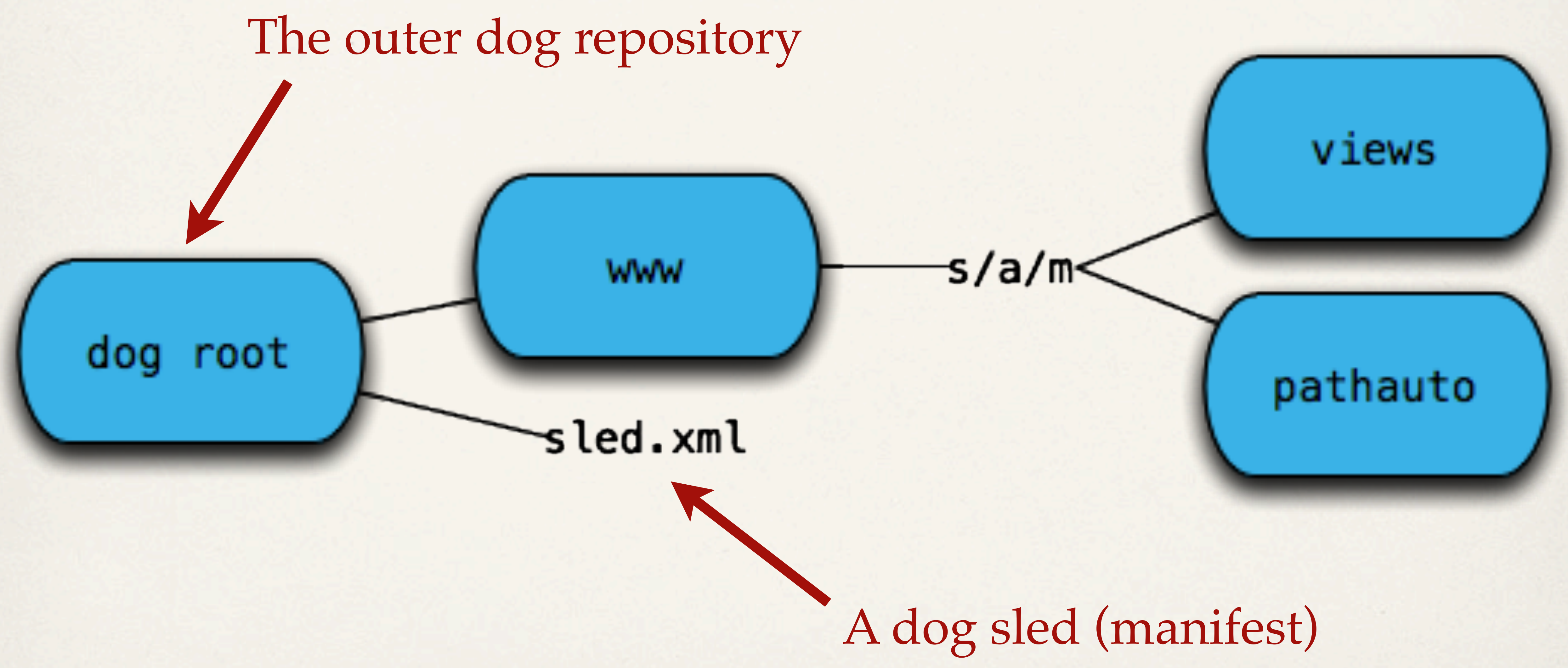
* DOG HOUSE!

so this is what dog systems actually start to look like.

Especially as sites become more complex, it becomes a better and better idea to NOT have the root of your repository correspond to the webroot. There are often scripts (or maybe even tests – PHPUnit, Selenium!!), etc. that you need as part of the project, and they make no sense under the webroot.

So dog kicks out a parent repository to contain this. This repository has no upstream. And it's called the Dog House! muahahahaha

Good dogs need houses. And sleds.



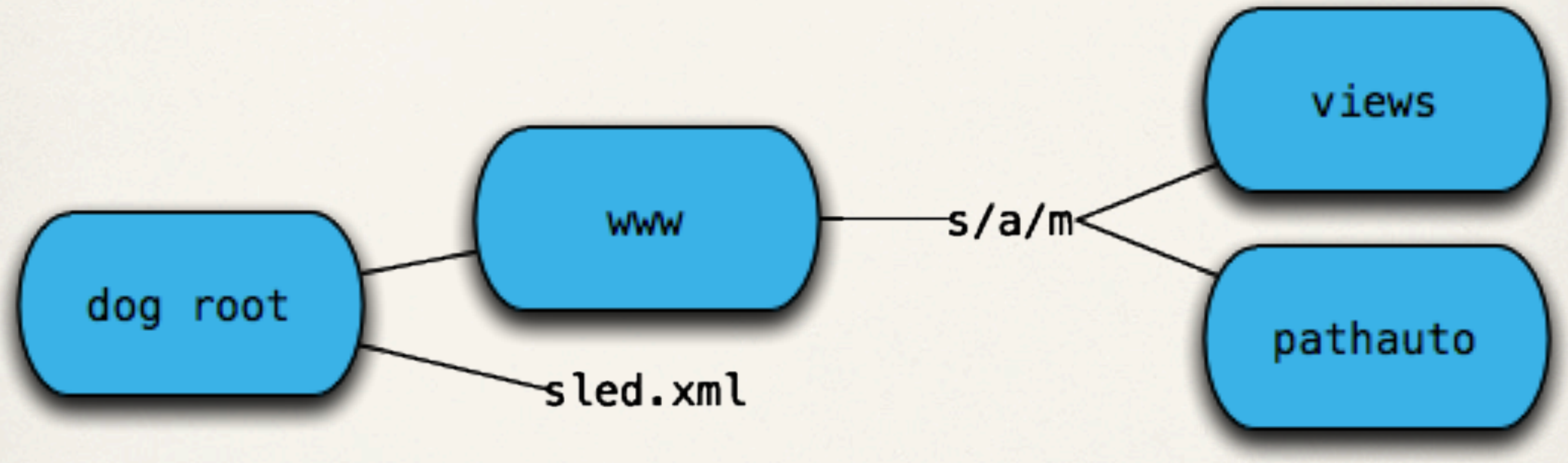
* DOG HOUSE!

so this is what dog systems actually start to look like.

Especially as sites become more complex, it becomes a better and better idea to NOT have the root of your repository correspond to the webroot. There are often scripts (or maybe even tests – PHPUnit, Selenium!!), etc. that you need as part of the project, and they make no sense under the webroot.

So dog kicks out a parent repository to contain this. This repository has no upstream. And it’s called the Dog House! muahahahaha

“Overall state” means database

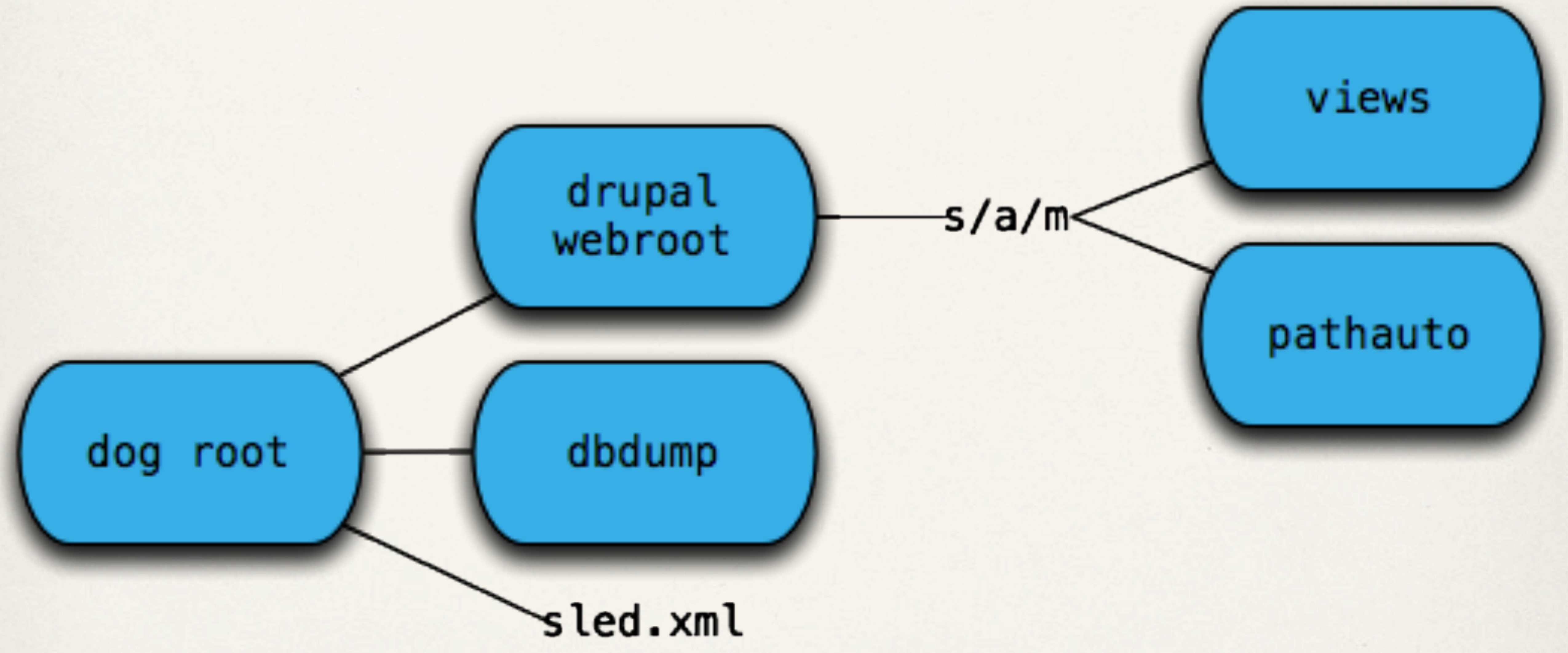


if dog is going to claim to represent the entire state of a drupal site, it’ll need to include some database data. ok, no problem – that’s just another repository.

and, as is hopefully becoming clear, dog’s skill area is really in managing clusters of git repositories.

why isolate db dumps into their own repo?
 * well, ya don’t have to!
 * it keeps the other repos more performant as they perform their operations as git can get kinda chokey on large files
 * individual repos also become ‘logic points’ in git. it’s easier for us to write generically helpful scripts around a base repo location, then have that arbitrarily located just “anywhere” in the tree.

“Overall state” means database

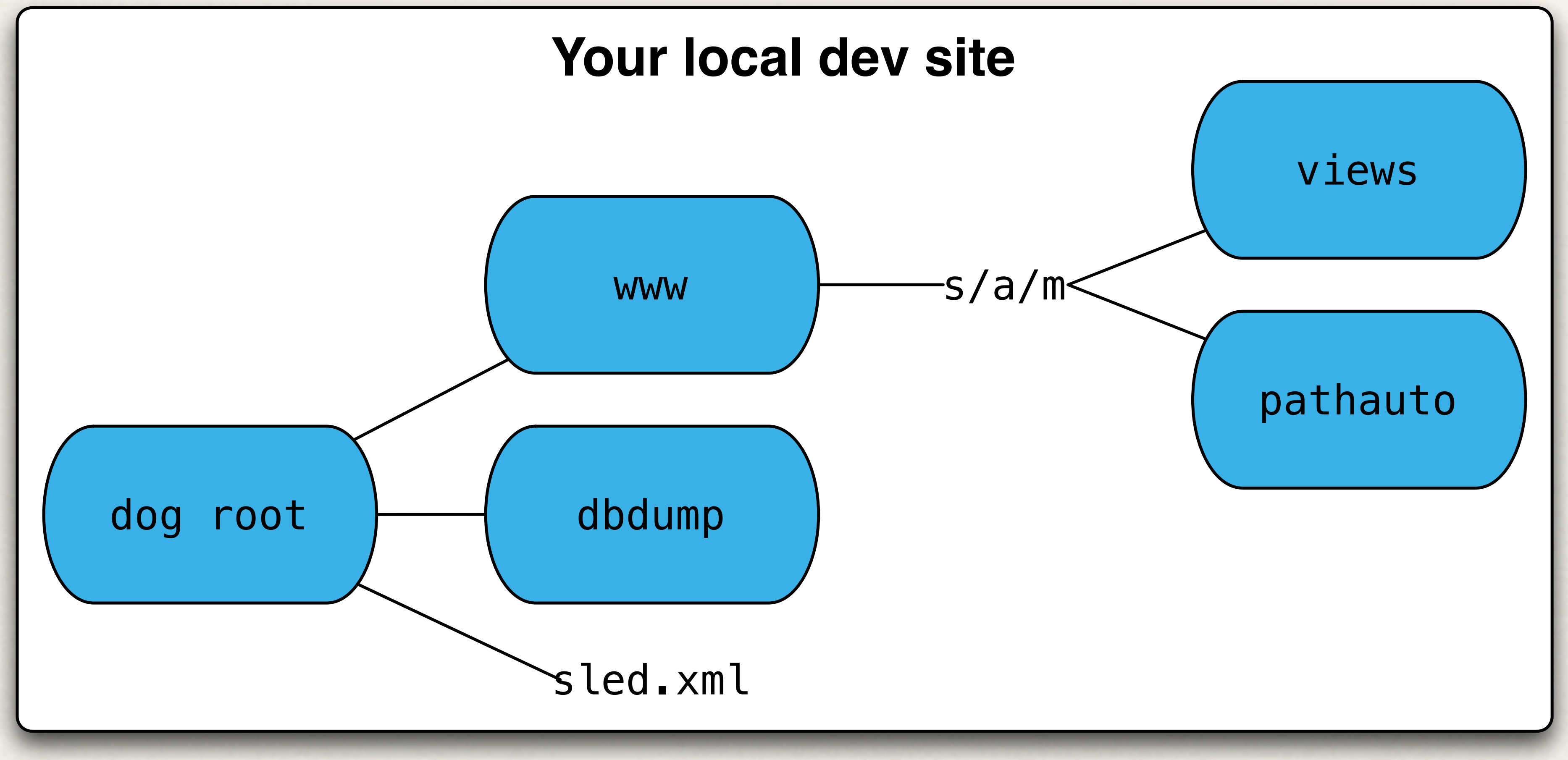


if dog is going to claim to represent the entire state of a drupal site, it’ll need to include some database data. ok, no problem – that’s just another repository.

and, as is hopefully becoming clear, dog’s skill area is really in managing clusters of git repositories.

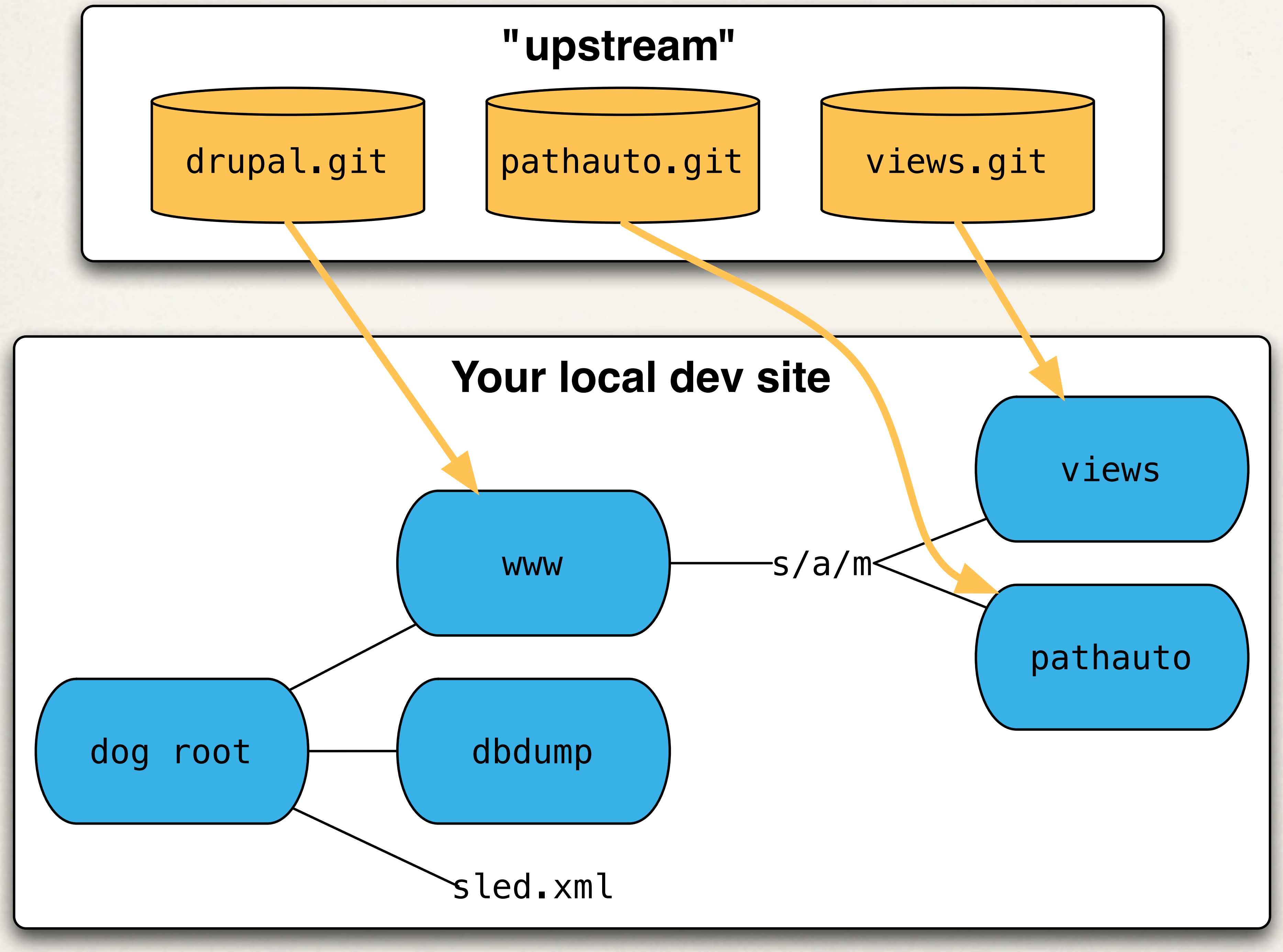
why isolate db dumps into their own repo?
 * well, ya don’t have to!
 * it keeps the other repos more performant as they perform their operations as git can get kinda chokey on large files
 * individual repos also become ‘logic points’ in git. it’s easier for us to write generically helpful scripts around a base repo location, then have that arbitrarily located just “anywhere” in the tree.

The remote jungle



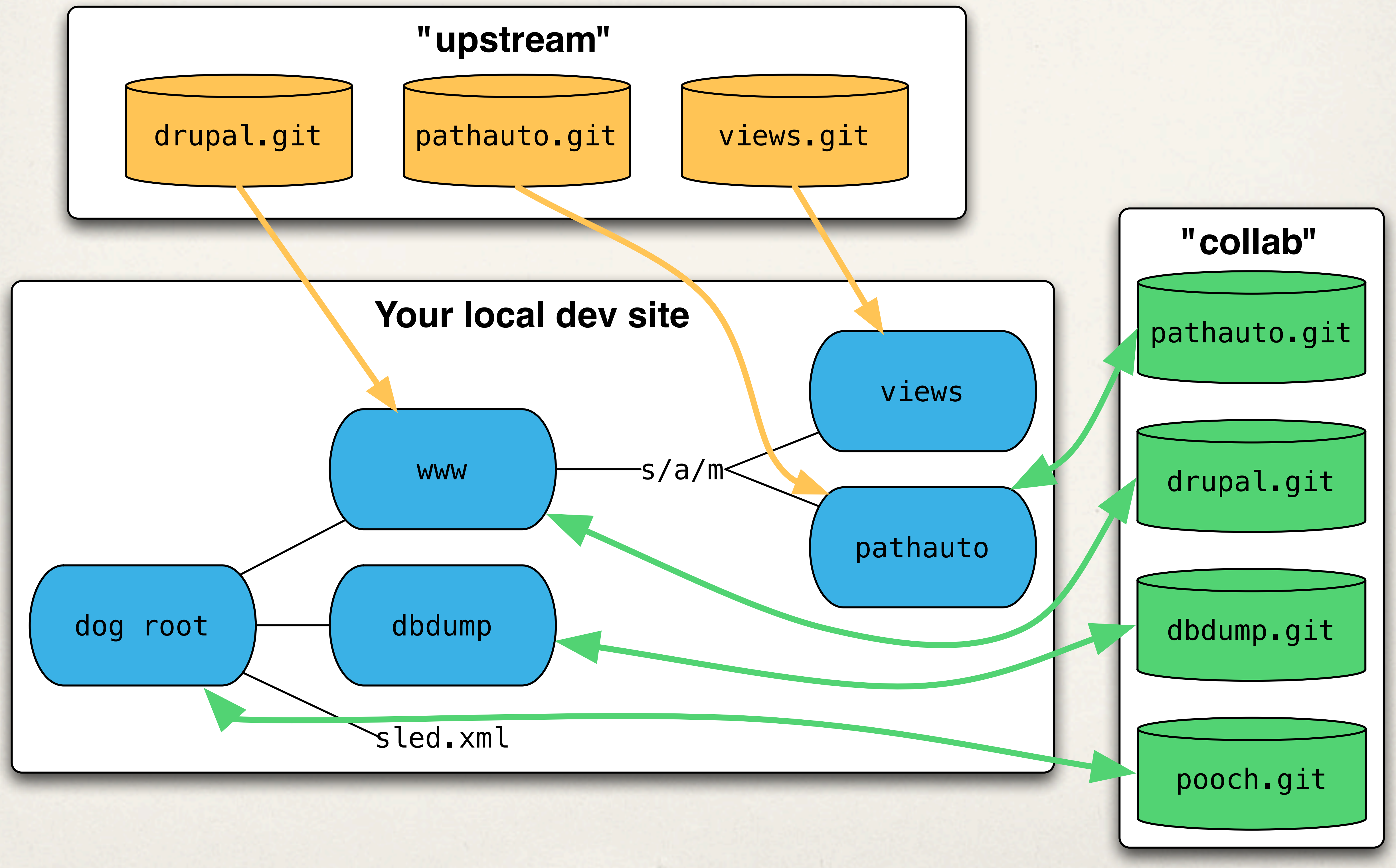
* collab has interesting potential for shops - you may need need a collab repo PER individual project. sharing them could be cool.

The remote jungle



* collab has interesting potential for shops - you may need need a collab repo PER individual project. sharing them could be cool.

The remote jungle

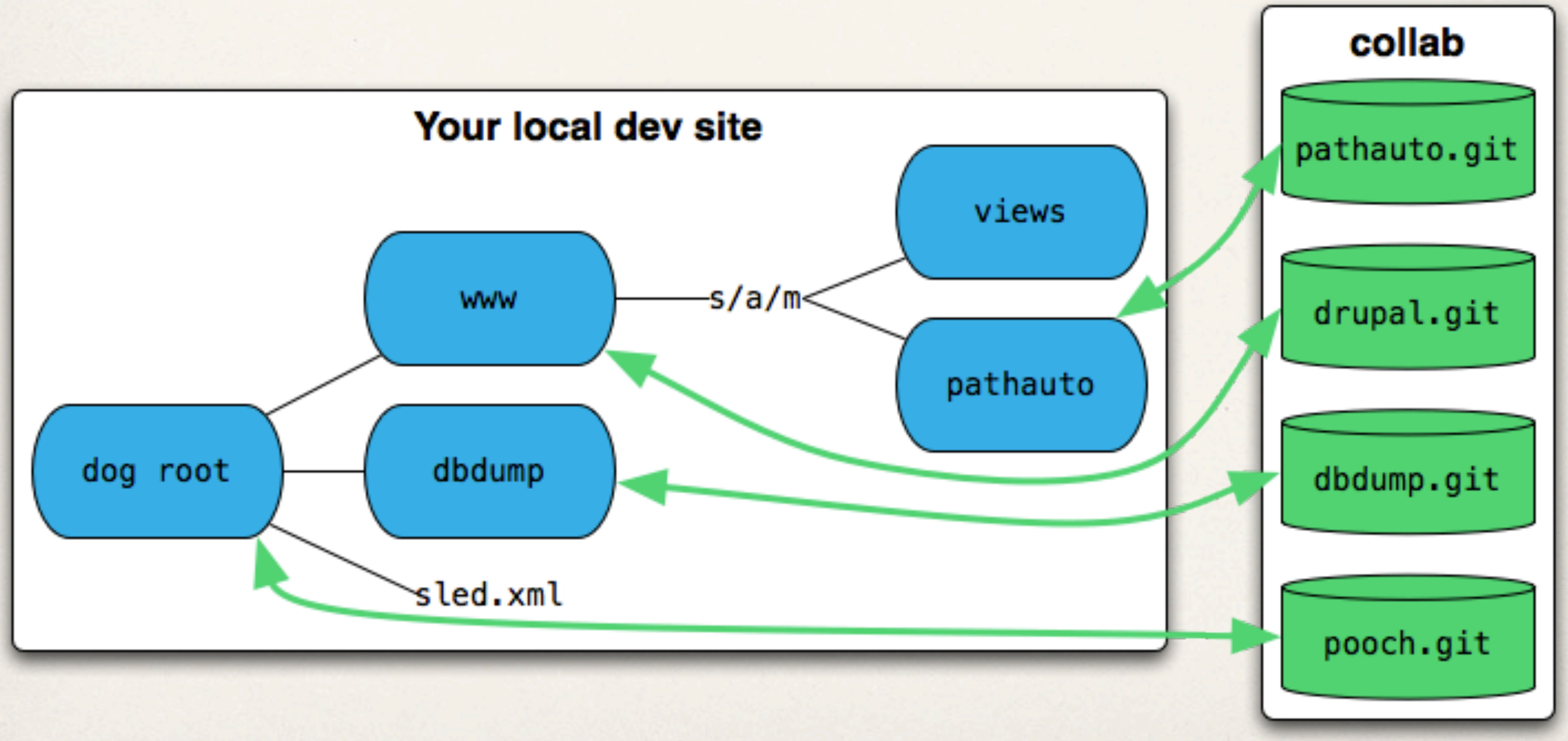


* collab has interesting potential for shops – you may need need a collab repo PER individual project. sharing them could be cool.

Pass it ‘round: rollup and rollout

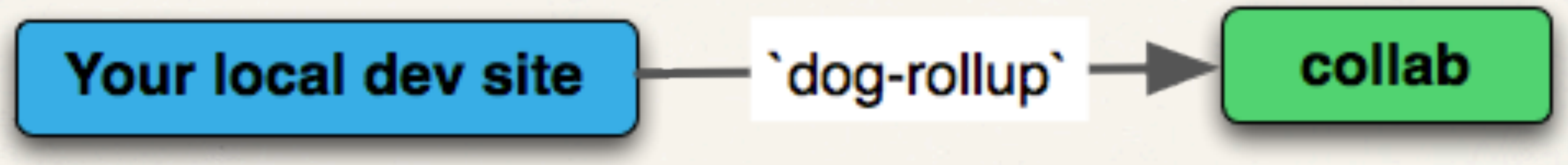
i hope this looks like a nightmare. honestly, managing nested git repositories IS a bit of a nightmare. which is why we have dog - because it goes from multiple commands, multiple checks, to just a single command set.

Pass it 'round: rollup and rollout



i hope this looks like a nightmare. honestly, managing nested git repositories IS a bit of a nightmare. which is why we have dog – because it goes from multiple commands, multiple checks, to just a single command set.

Pass it ‘round: rollup and rollout

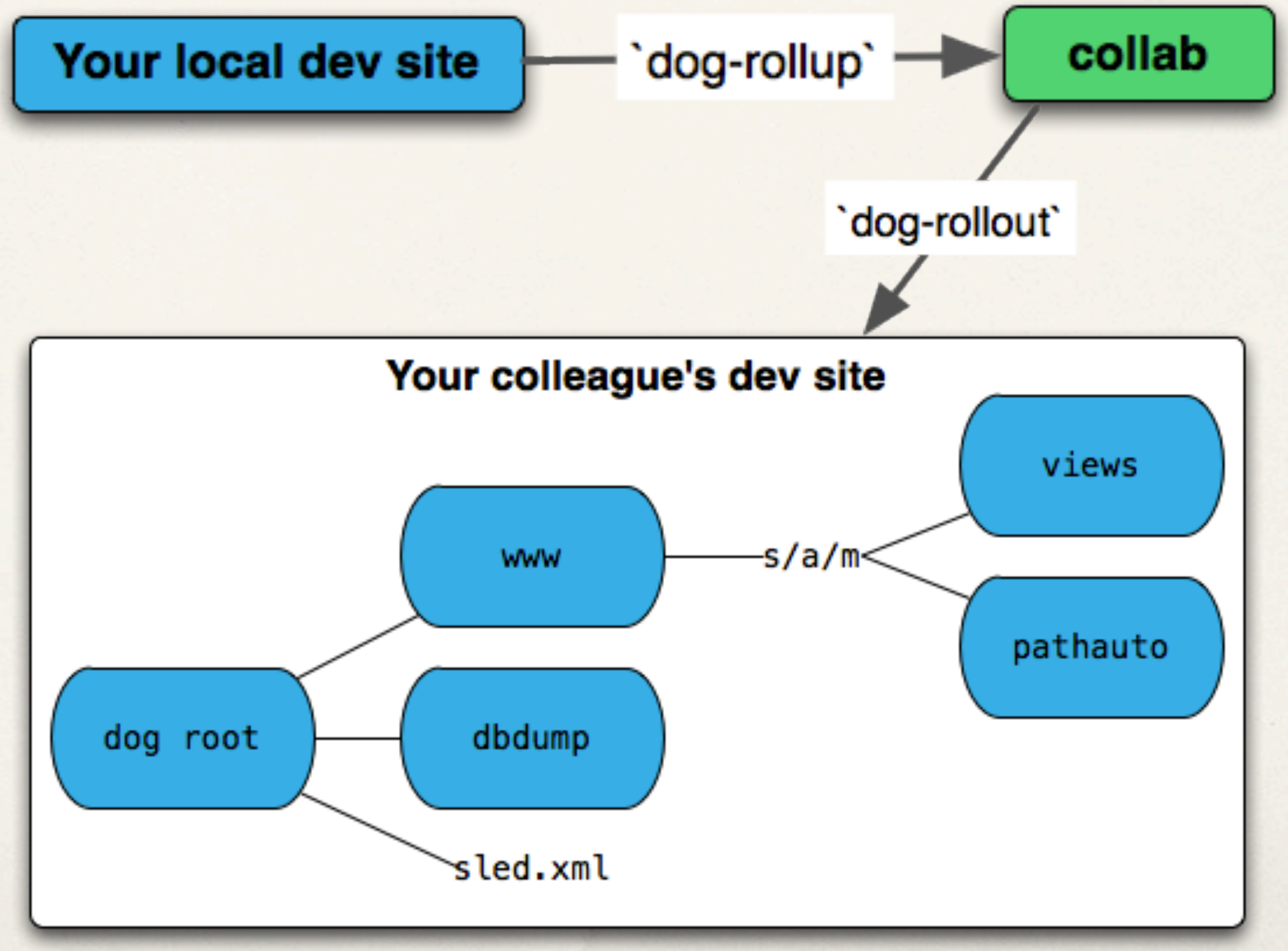


dog treats the entire cluster of git repositories as a single entity, and manipulates it as a whole. so you run one command - dog-rollup - from your local dev site to take the latest code, database, etc. and push it all up into collab. and from there

<transition>

it’s just one more command to provision out a new instance of the site.

Pass it 'round: rollup and rollout

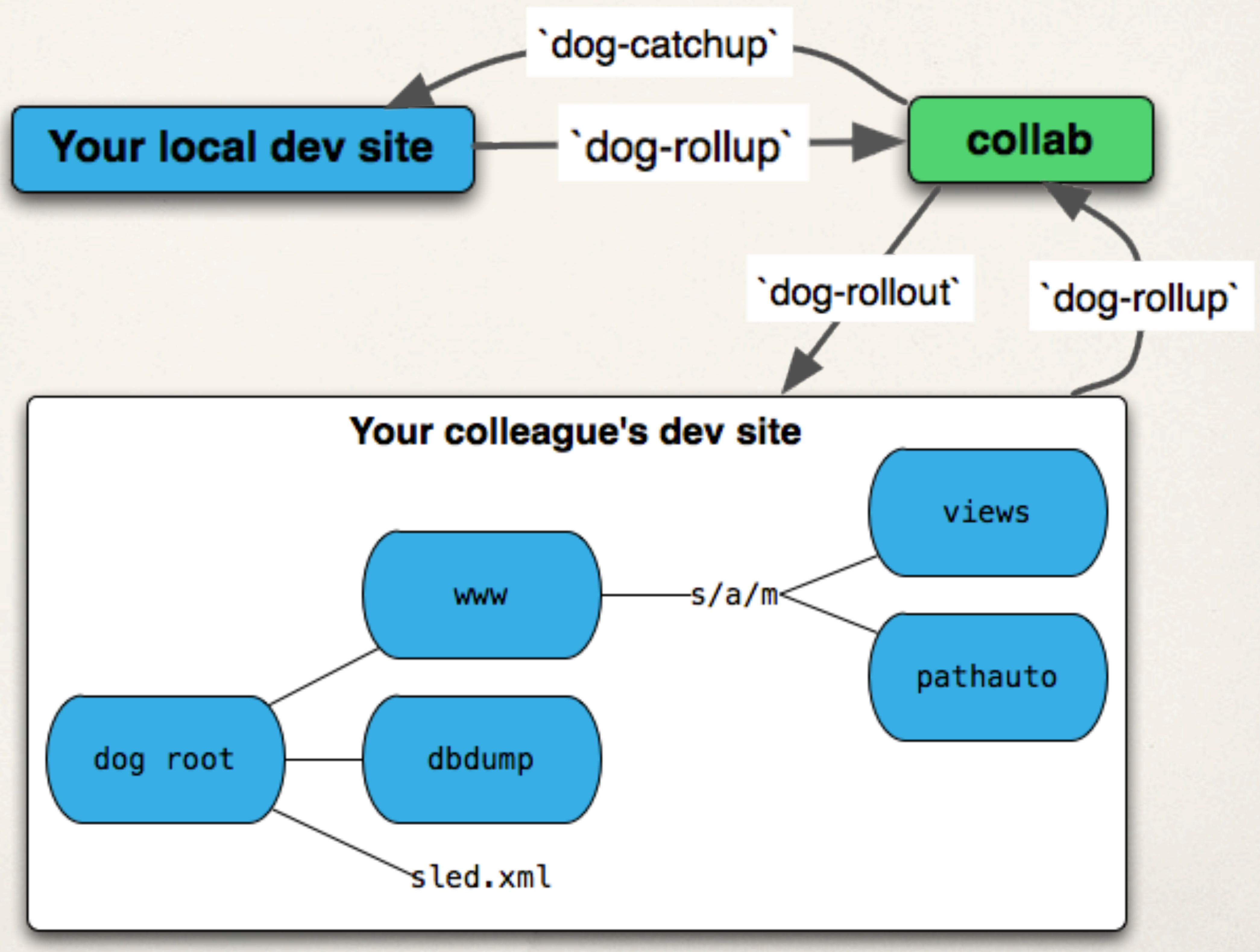


dog treats the entire cluster of git repositories as a single entity, and manipulates it as a whole. so you run one command – dog-rollup – from your local dev site to take the latest code, database, etc. and push it all up into collab. and from there

<transition>

it's just one more command to provision out a new instance of the site.

Pass it ‘round: rollup and rollout

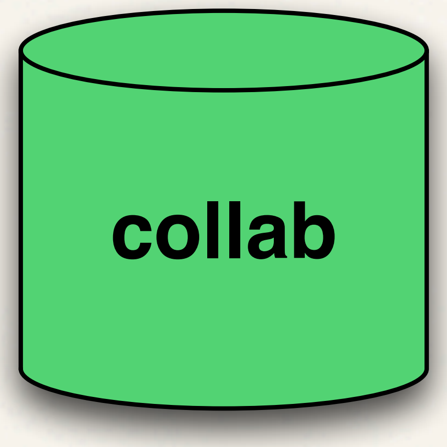


dog treats the entire cluster of git repositories as a single entity, and manipulates it as a whole. so you run one command – dog-rollup – from your local dev site to take the latest code, database, etc. and push it all up into collab. and from there

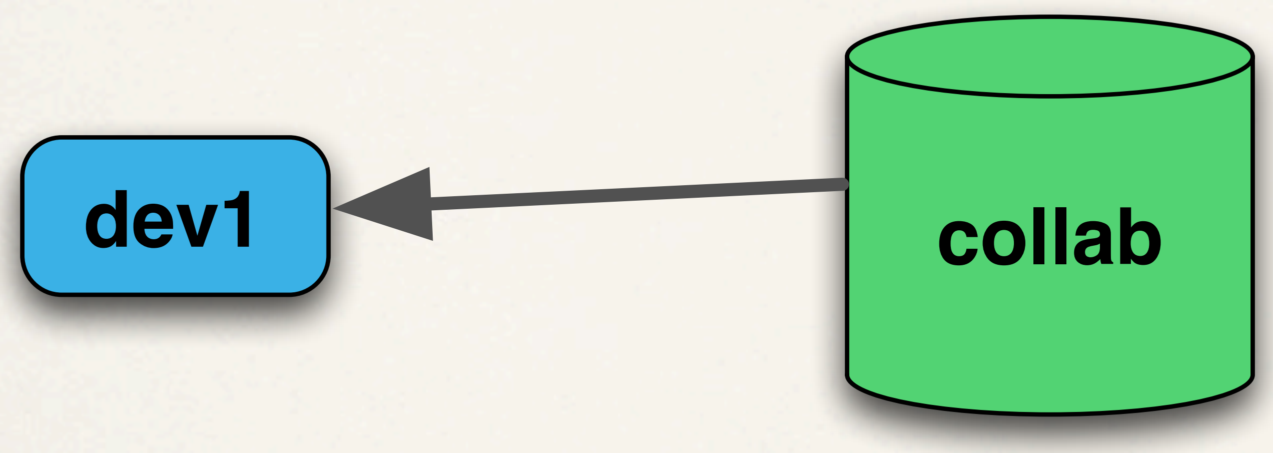
<transition>

it’s just one more command to provision out a new instance of the site.

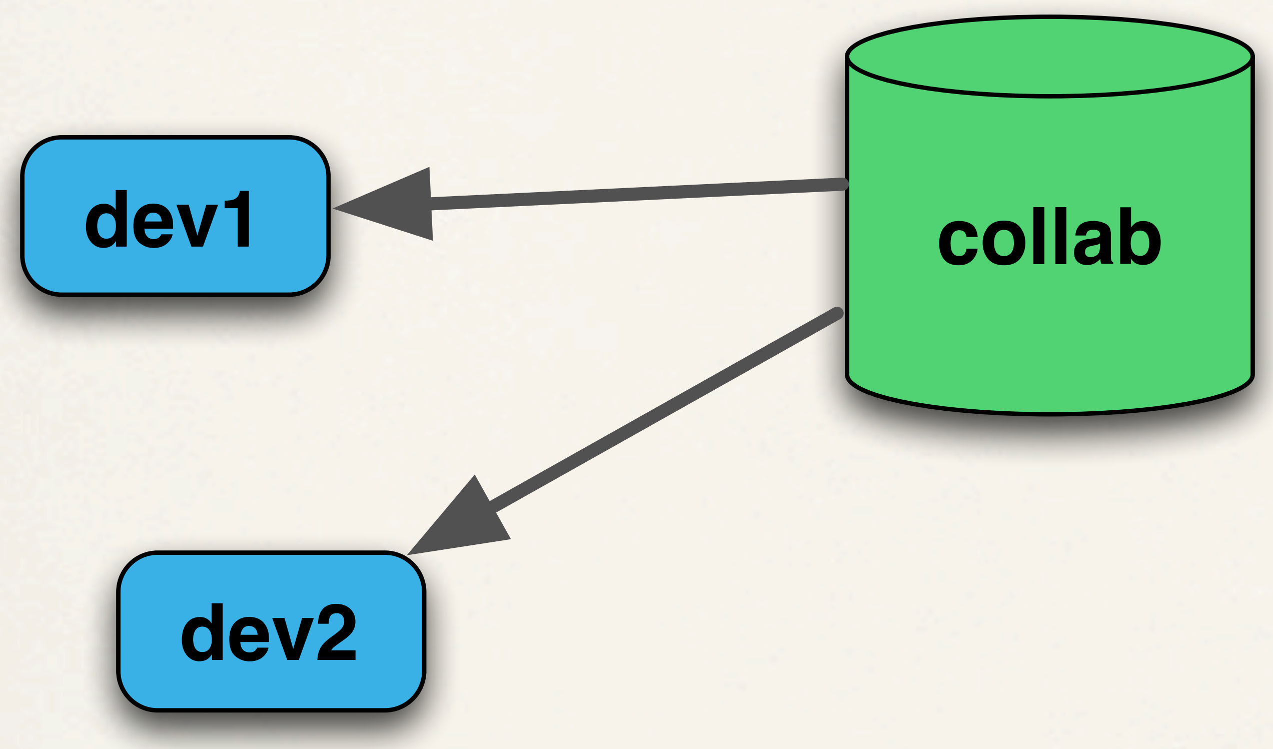
...and out, and out...



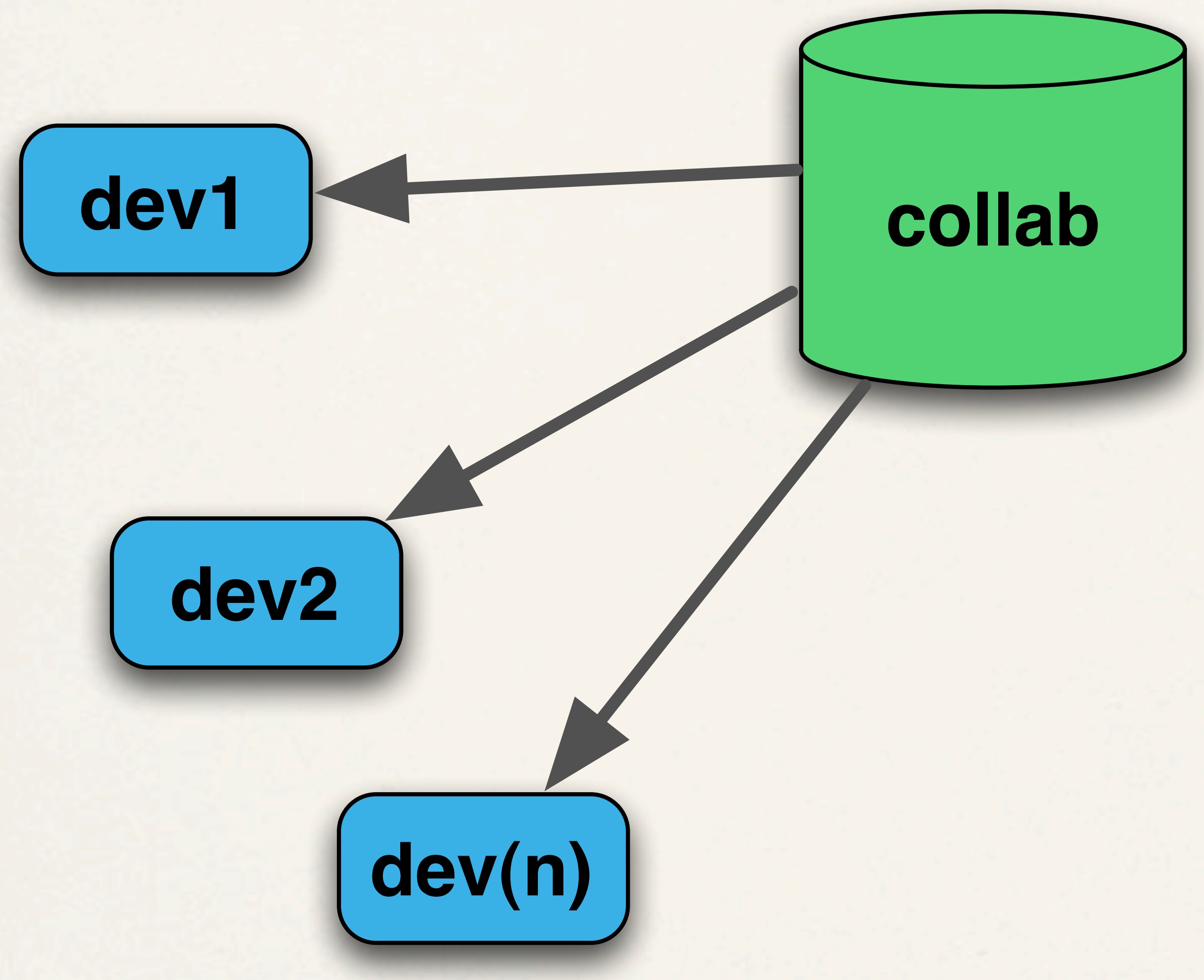
...and out, and out...



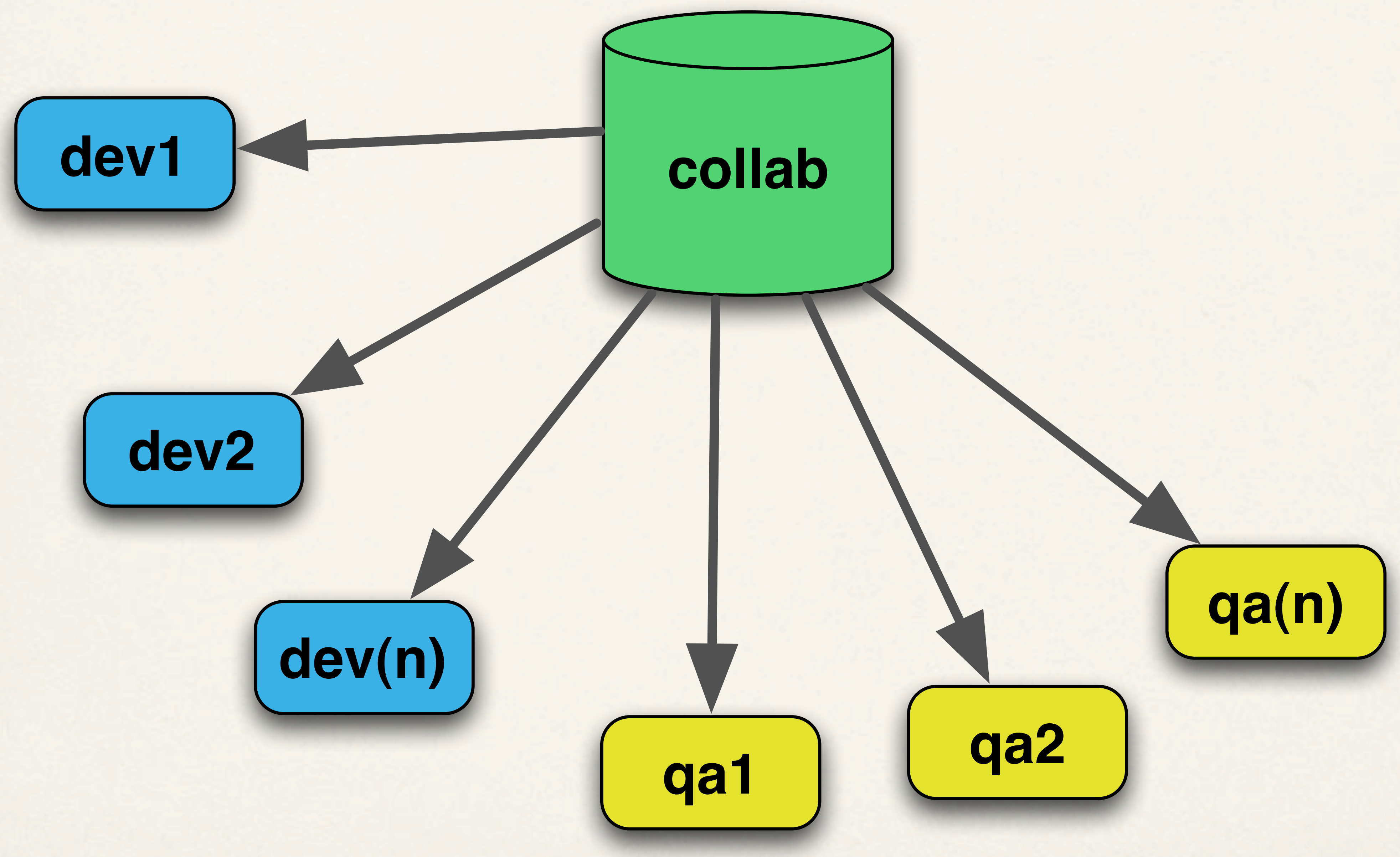
...and out, and out...



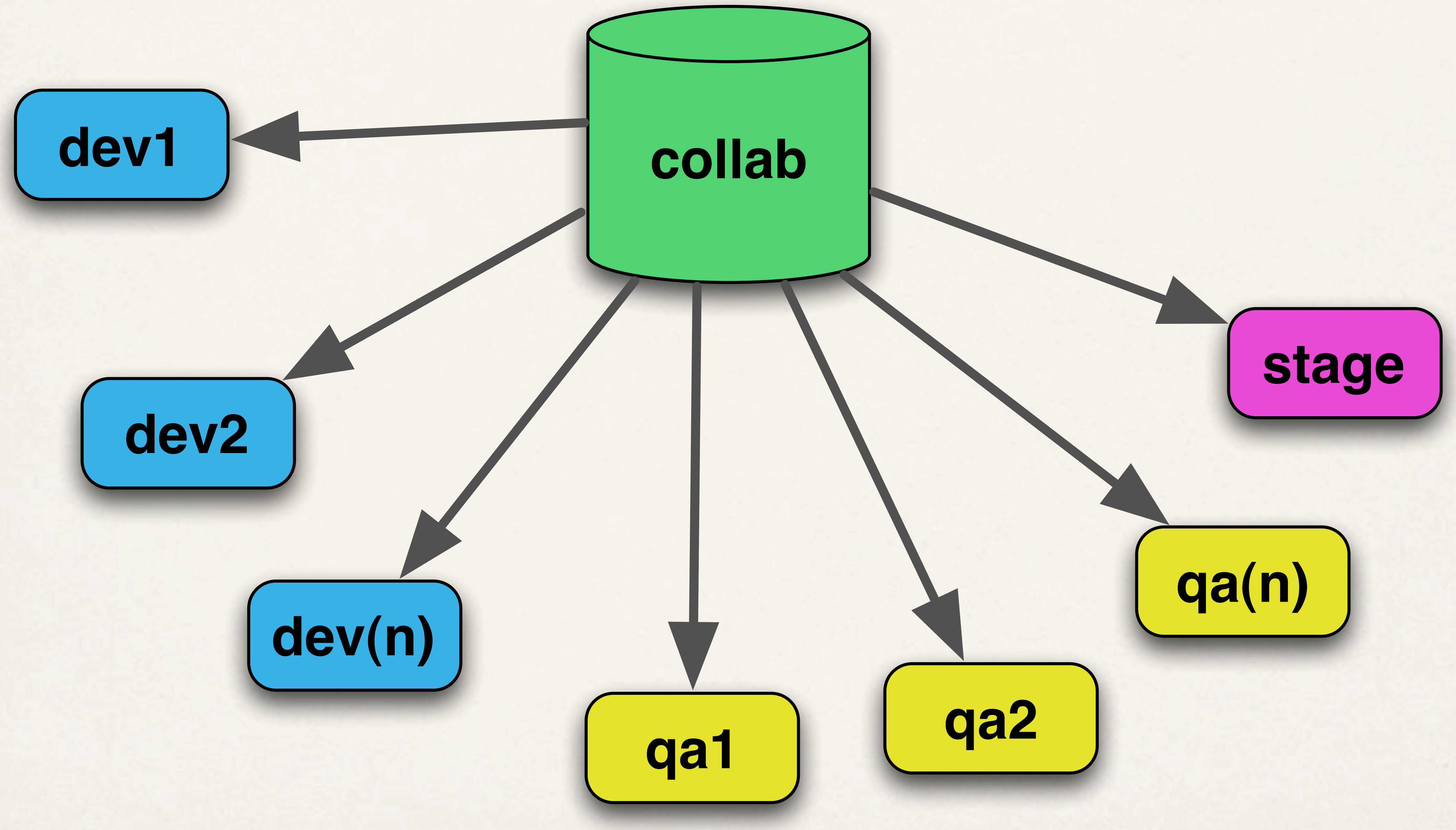
...and out, and out...



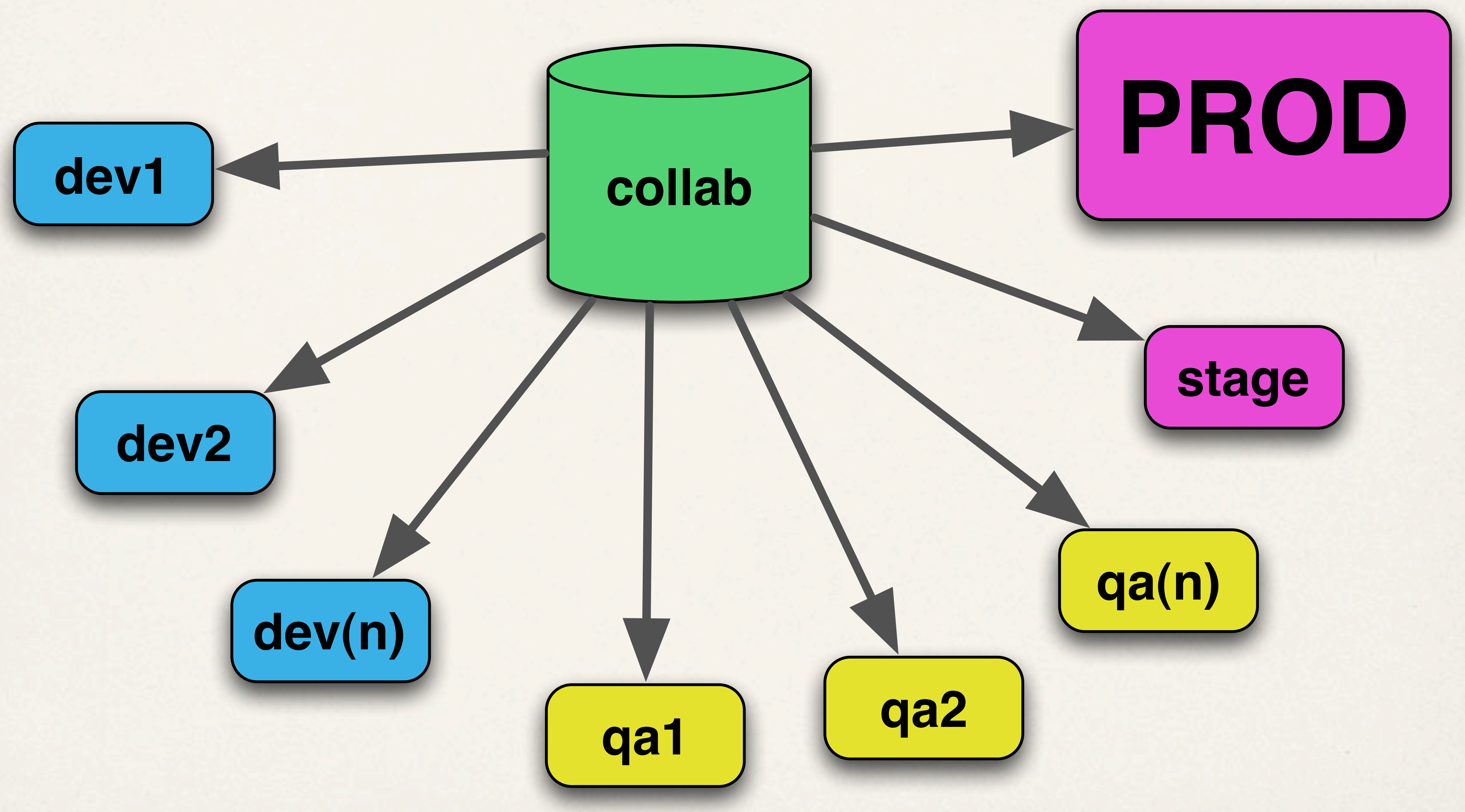
...and out, and out...



...and out, and out...



...and out, and out...



RAAAAGGHHH!!

The day-to-day commands

* init interactively prompts you for db settings, other localized information
* dog-dl is equivalent to regular dl, just does it...doggy style
*

The day-to-day commands

- `drush dog-init`

* init interactively prompts you for db settings, other localized information
* dog-dl is equivalent to regular dl, just does it...doggy style
*

The day-to-day commands

- `drush dog-init`
- `drush dog-dl`

* init interactively prompts you for db settings, other localized information
* dog-dl is equivalent to regular dl, just does it...doggy style
*

The day-to-day commands

- `drush dog-init`
- `drush dog-dl`
- `drush dog-{rollup,rollout,catchup}`

* init interactively prompts you for db settings, other localized information
* dog-dl is equivalent to regular dl, just does it...doggy style
*

The day-to-day commands

- `drush dog-init`
- `drush dog-dl`
- `drush dog-{rollup,rollout,catchup}`
- `git {commit,checkout,branch,tag,add,rm}`

* init interactively prompts you for db settings, other localized information
* dog-dl is equivalent to regular dl, just does it...doggy style
*

Controlling build commands

 * build system: take some set of raw data + configuration and process it, generally producing a bunch more

 * meta-deployment because it’s more like a build container for other strategies you might slot in - deploy module, features, other such things.

Controlling build commands

- **`dog-rollup`**, **`dog-rollout`**, and **`dog-catchup`** are the dog commands that are most like build systems

 * build system: take some set of raw data + configuration and process it, generally producing a bunch more

 * meta-deployment because it’s more like a build container for other strategies you might slot in - deploy module, features, other such things.

Controlling build commands

- **`dog-rollup`, `dog-rollout`,** and **`dog-catchup`** are the dog commands that are most like build systems
- We CANNOT hope to make dog match your special-snowflake usecase out of the box with the build tasks. No way.

* build system: take some set of raw data + configuration and process it, generally producing a bunch more

* meta-deployment because it’s more like a build container for other strategies you might slot in - deploy module, features, other such things.

Controlling build commands

- **`dog-rollup`**, **`dog-rollout`**, and **`dog-catchup`** are the dog commands that are most like build systems
- We CANNOT hope to make dog match your special-snowflake usecase out of the box with the build tasks. No way.
- So here, we punt to a build system (Phing, Phake, Pake - not yet decided) to do the actual work

 * build system: take some set of raw data + configuration and process it, generally producing a bunch more

 * meta-deployment because it's more like a build container for other strategies you might slot in - deploy module, features, other such things.

Controlling build commands

- **`dog-rollup`, `dog-rollout`,** and **`dog-catchup`** are the dog commands that are most like build systems
- We CANNOT hope to make dog match your special-snowflake usecase out of the box with the build tasks. No way.
- So here, we punt to a build system (Phing, Phake, Pake - not yet decided) to do the actual work
    - dog's API is available, but it's mostly in *your* hands to determine the details of your rollup, rollout, and catchup process

* build system: take some set of raw data + configuration and process it, generally producing a bunch more

* meta-deployment because it's more like a build container for other strategies you might slot in - deploy module, features, other such things.

Controlling build commands

- **`dog-rollup`, `dog-rollout`,** and **`dog-catchup`** are the dog commands that are most like build systems
- We CANNOT hope to make dog match your special-snowflake usecase out of the box with the build tasks. No way.
- So here, we punt to a build system (Phing, Phake, Pake - not yet decided) to do the actual work
  - dog's API is available, but it's mostly in *your* hands to determine the details of your rollup, rollout, and catchup process
- Also, this is why dog is not a "deployment" system; more a meta-deployment system

 * build system: take some set of raw data + configuration and process it, generally producing a bunch more

 * meta-deployment because it's more like a build container for other strategies you might slot in - deploy module, features, other such things.

Environments/Build Modes

Environments/Build Modes

- If dog is running dev, test, stage, and prod, then it will need to be able to reflect some differences

Environments/Build Modes

- If dog is running dev, test, stage, and prod, then it will need to be able to reflect some differences
  - Don’t send emails from dev sites!

Environments/Build Modes

- If dog is running dev, test, stage, and prod, then it will need to be able to reflect some differences
  - Don’t send emails from dev sites!
  - Ensuring a module is enabled/disabled

Environments/Build Modes

- If dog is running dev, test, stage, and prod, then it will need to be able to reflect some differences
  - Don’t send emails from dev sites!
  - Ensuring a module is enabled/disabled
- This is something we want, though right now I’m thinking 2.0

dog-as-package

dog-as-package

- Since dog is designed to capture the overall state of a Drupal site, it’s ideal to integrate with tools that might make use of such a package

dog-as-package

- Since dog is designed to capture the overall state of a Drupal site, it's ideal to integrate with tools that might make use of such a package
- Aegir could use it as an alternative "container" to Drush Make for rolling out platforms. Then you could actually hack on them!

dog-as-package

- Since dog is designed to capture the overall state of a Drupal site, it's ideal to integrate with tools that might make use of such a package
- Aegir could use it as an alternative "container" to Drush Make for rolling out platforms. Then you could actually hack on them!
- Drupally hosting providers (Pantheon, Dev Cloud, etc.) could build on dog to do a variety of things

dog-as-package

- Since dog is designed to capture the overall state of a Drupal site, it's ideal to integrate with tools that might make use of such a package
- Aegir could use it as an alternative "container" to Drush Make for rolling out platforms. Then you could actually hack on them!
- Drupally hosting providers (Pantheon, Dev Cloud, etc.) could build on dog to do a variety of things
- I can imagine a dog provider in puppet that plops down a dog instance at a particular location

dog-as-package

- Since dog is designed to capture the overall state of a Drupal site, it’s ideal to integrate with tools that might make use of such a package
- Aegir could use it as an alternative “container” to Drush Make for rolling out platforms. Then you could actually hack on them!
- Drupally hosting providers (Pantheon, Dev Cloud, etc.) could build on dog to do a variety of things
- I can imagine a dog provider in puppet that plops down a dog instance at a particular location
- If that makes you squirm, dog will provide an ‘export’ mode, which could be used to write flat rpms, debs, for use in cfg mgmt tools

What did you think?

DRUPALCON
LONDON

What did you think?

Locate this session on the DrupalCon London website:

http://london2011.drupal.org/conference/schedule

DRUPALCON LONDON

What did you think?

Locate this session on the DrupalCon London website:

http://london2011.drupal.org/conference/schedule

Click the “Take the survey” link

DRUPALCON LONDON

What did you think?

Locate this session on the DrupalCon London website:

http://london2011.drupal.org/conference/schedule

Click the “Take the survey” link

THANK YOU!

DRUPALCON LONDON