Testing and Promoting Infrastructure and Configuration Changes

Tools and patterns are rapidly evolving in this area so this will be a discussion session, not coding.

In what ways can you test as you are writing puppet code?

Testing Puppet

- `puppet apply --noop db.pp`
- unit testing?
- MDD using Nagios checks
- with directly invoked Nagios checks
- with application functional tests
- base image vs production image
- implicit testing vs explicit testing
- test as part of CI

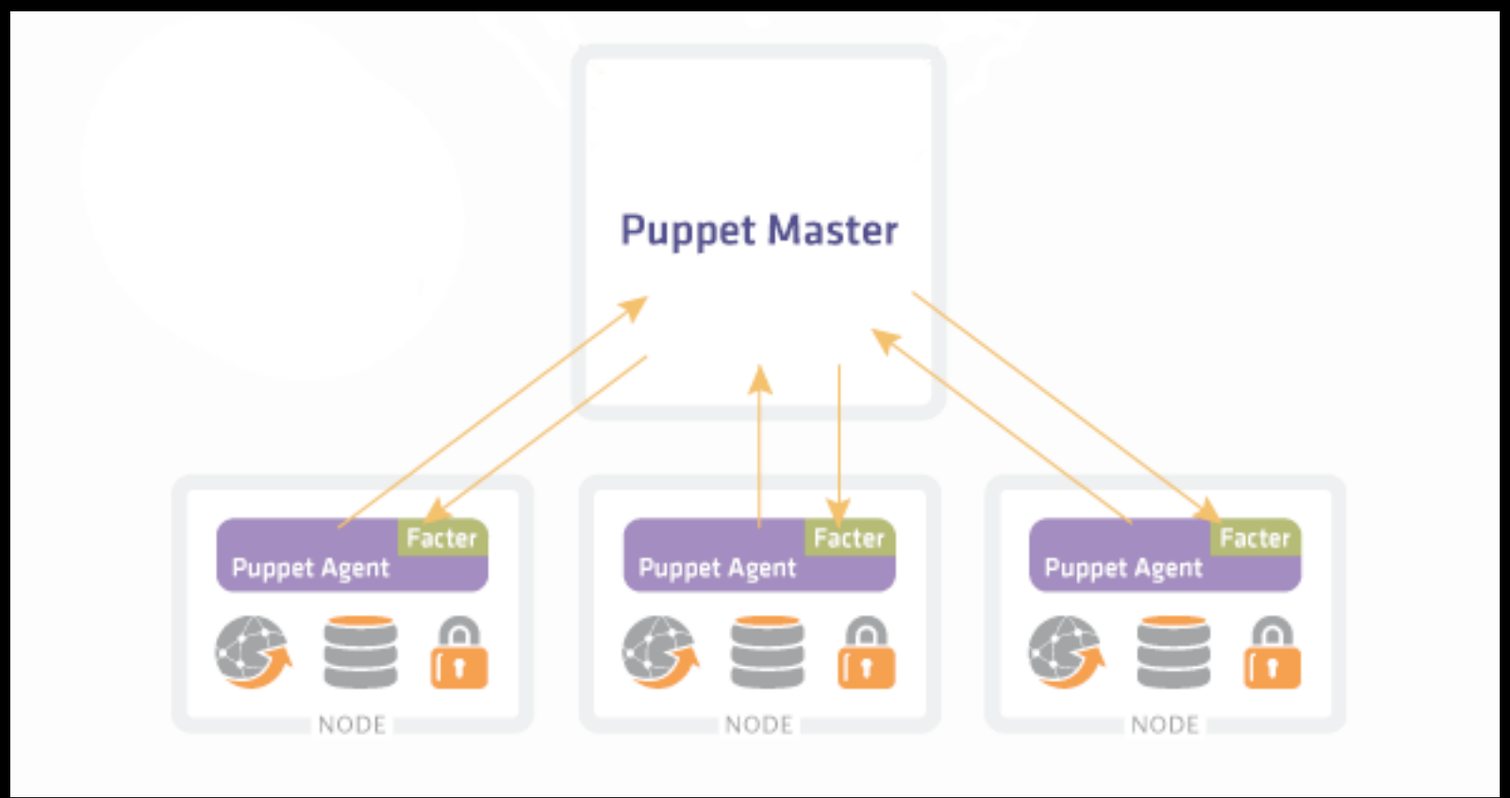
We've been manually executing puppet manifests on our 'production' nodes. This is clearly not a scalable solution. How would you orchestrate executing the manifests if you had 20 web nodes?

Orchestrating Puppet Execution

- bash + scp + ssh
- Capistrano or similar
- Bladelogic, etc
- Go agents
- cron jobs
- Puppet Master
- ControlTier, RunDeck
- bittorrent

Puppet Master

- Centralized management of manifests
- Agents on each node regularly ask Master for manifests to apply



Puppet Master

Pros

- Centralized tracking - what is doing what
- Centralized control/ orchestration
- Each nodes only knows about its manifests

Cons

- Another system to maintain
- Scalability concern (in the past?)
- Not a good story around versioning
- Not a good story around environments.

How would you promote changes to opencart (application and systems configuration) from a dev desktop through CI & test and into production?

You might want to draw a picture.

How can we use puppet to manage dev workstations and continuous integration?

How does the dev/ci puppet relate to puppet you use to manage production?

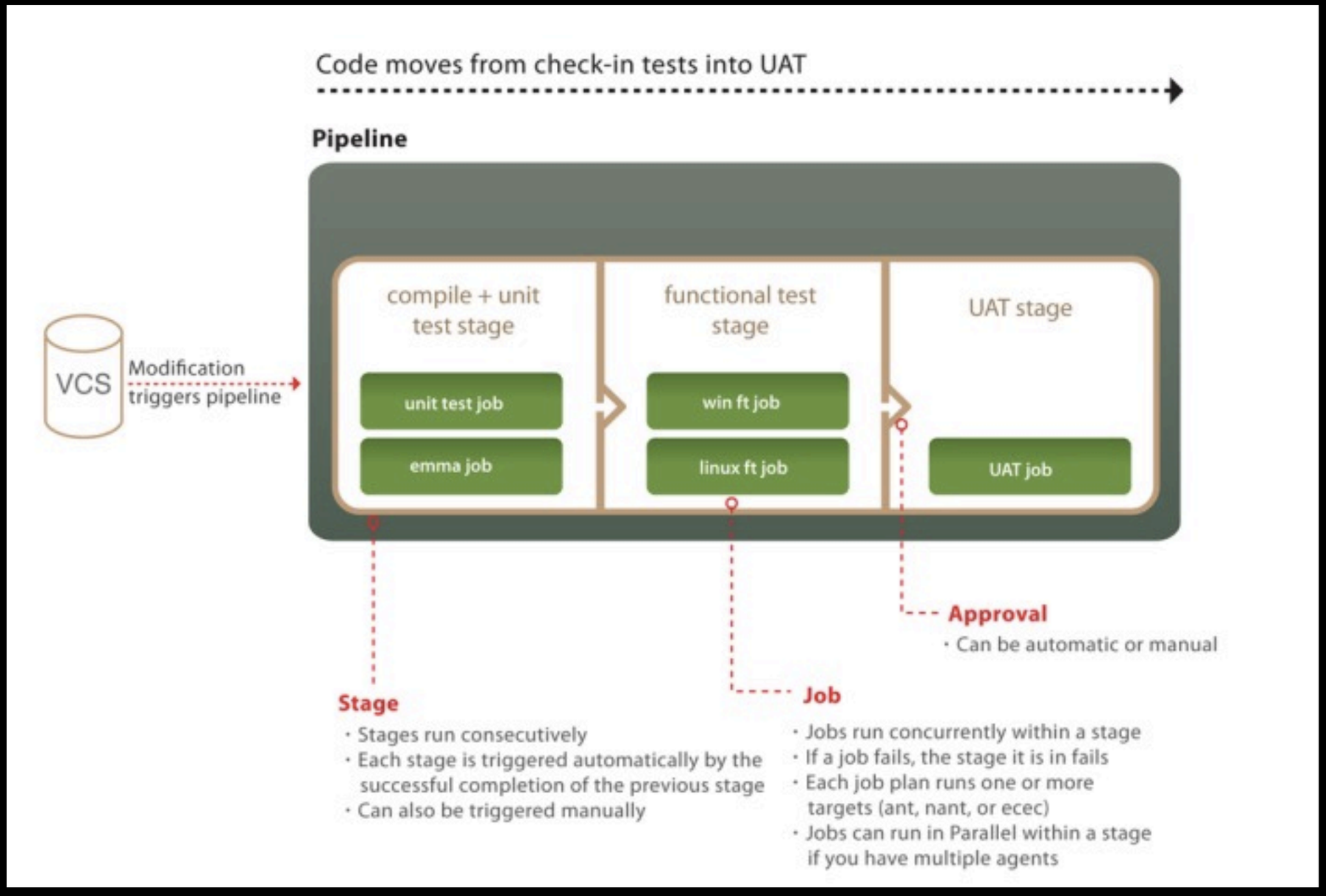
Would you promote changes to Nagios config in the pipeline?

How could we pass values like dbhost, dbpassword, etc to our web.pp puppet manifest?

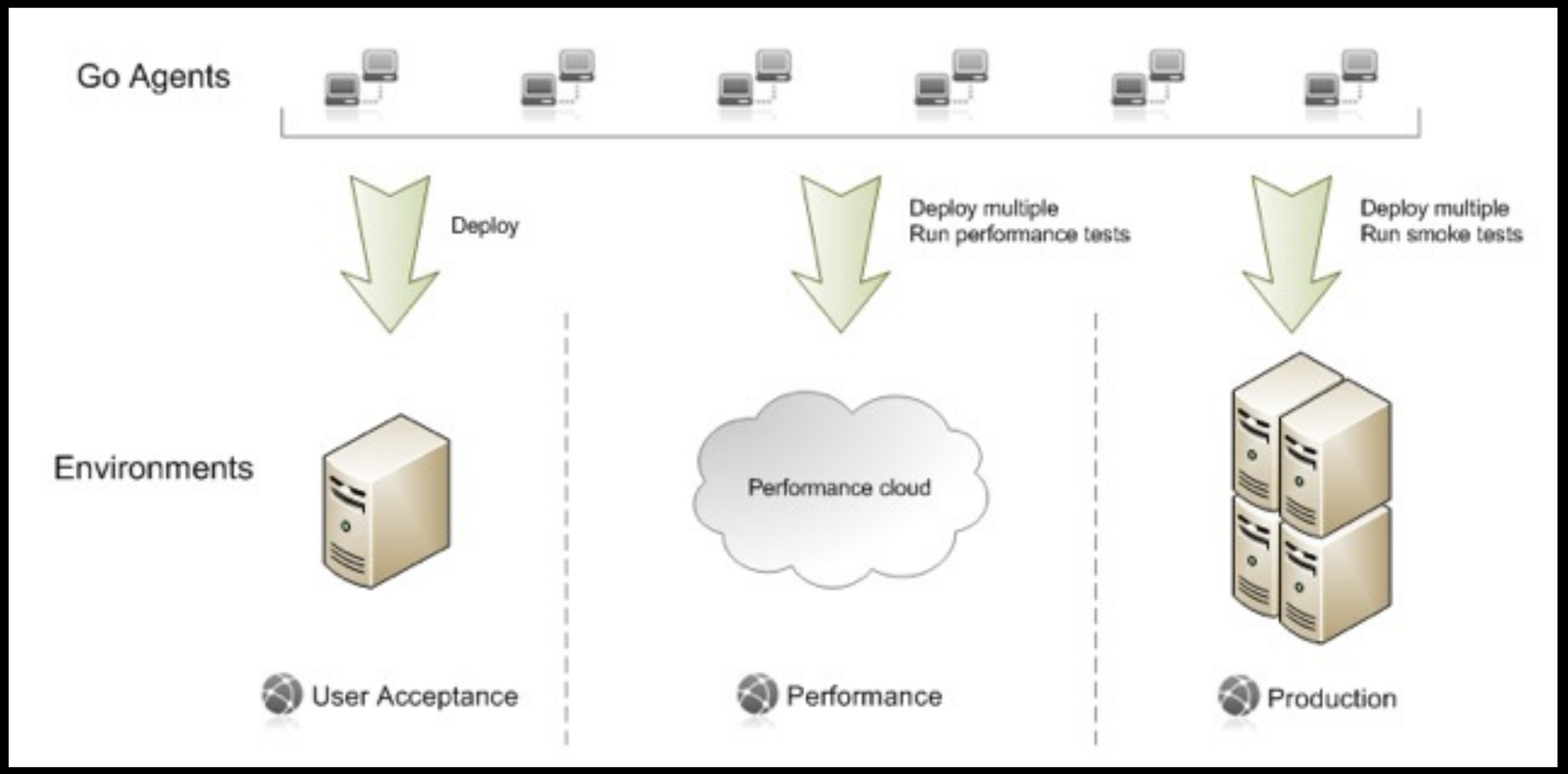
- PuppetMaster
- environment variable / facter
- hand-rolled function
  - like template()

Building a Pipeline for OpenCart with Go

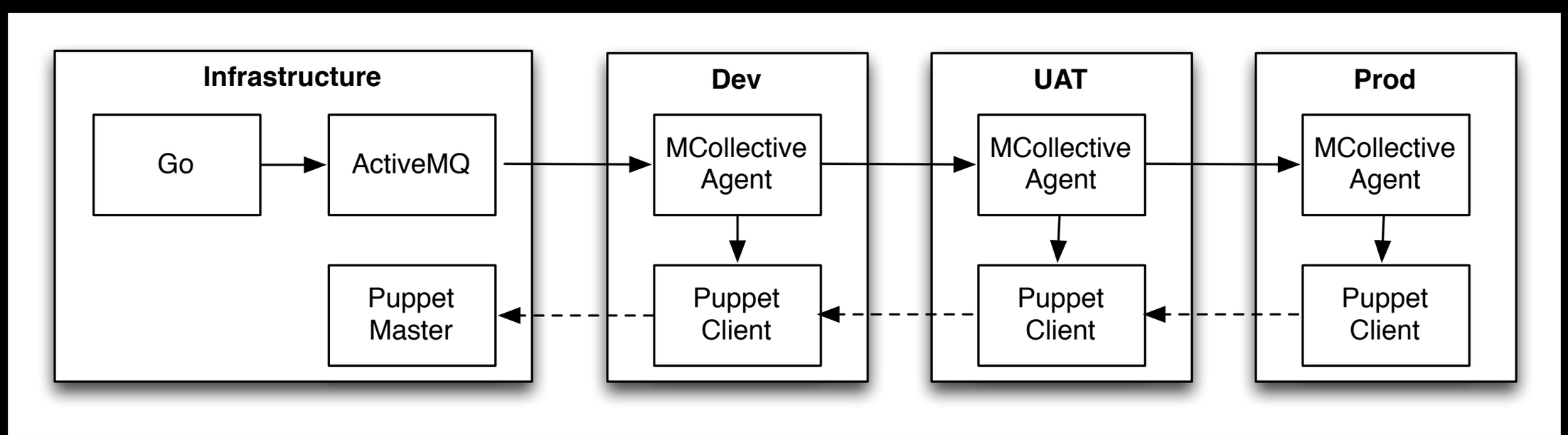
pipelines


Code moves from check-in tests into UAT
Pipeline
compile + unit test stage
functional test stage
UAT stage
unit test job
emma job
win ft job
linux ft job
UAT job
VCS
Modification triggers pipeline
Approval
• Can be automatic or manual
Stage
• Stages run consecutively
• Each stage is triggered automatically by the successful completion of the previous stage
• Can also be triggered manually
Job
• Jobs run concurrently within a stage
• If a job fails, the stage it is in fails
• Each job plan runs one or more targets (ant, nant, or ecec)
• Jobs can run in Parallel within a stage if you have multiple agents

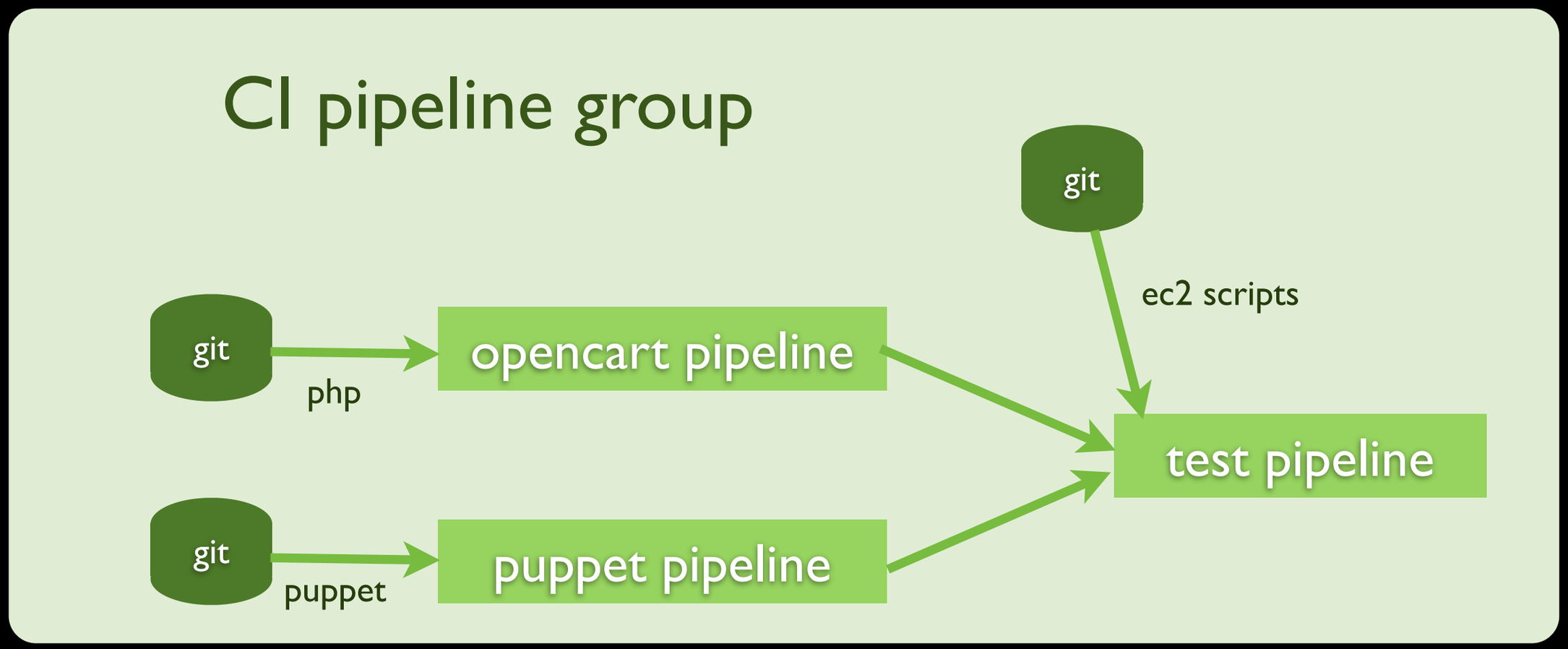
environments



or



demo of our go setup



Look at your instance

vagrant up go

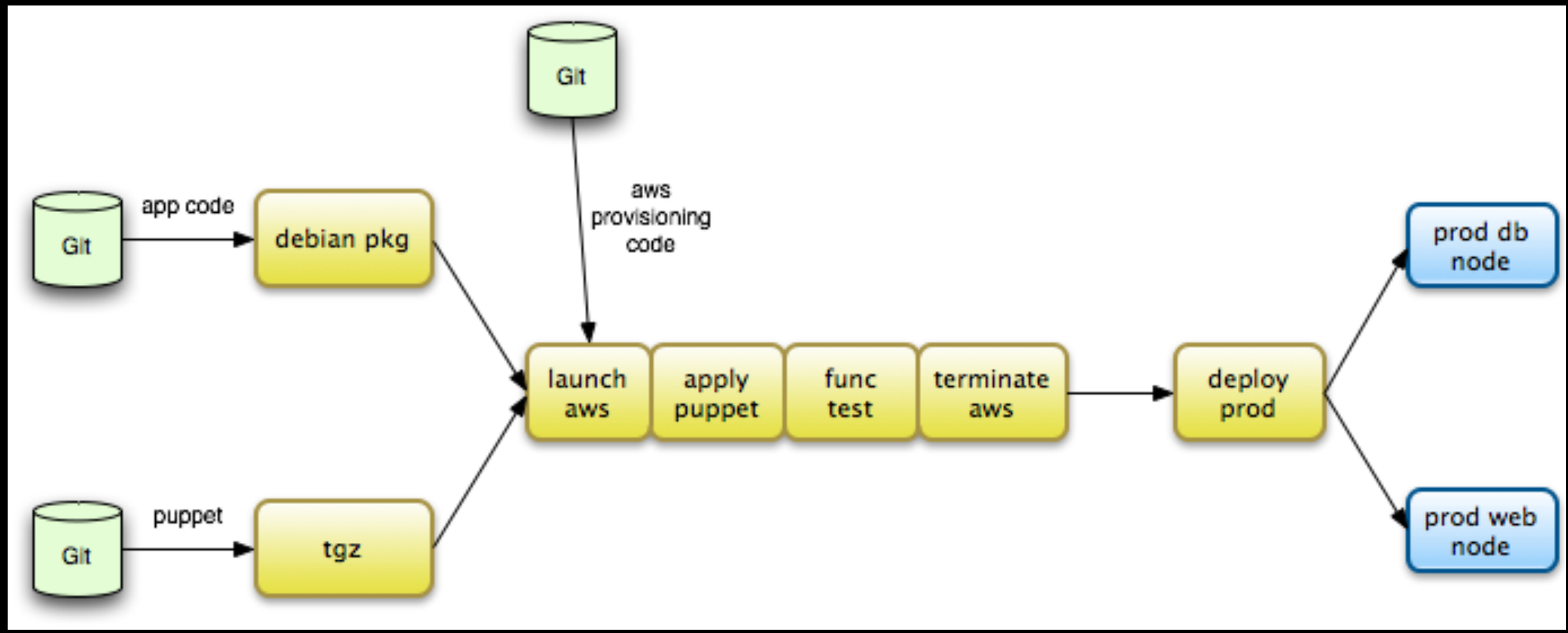
http://go:8153

on your laptop

- vagrant ssh-config go >> ~/.ssh/config
- git clone git@go:/var/git/puppet.git
- git clone git@go:/var/git/opencart.git
- git clone git@go:/var/git/build-scripts.git

questions

- What does the puppet pipeline do?
- For test pipeline
  - what does the launch stage do?
  - what does the configure stage do?
  - what does the test stage do?
  - what does the terminate stage do?
- Where do the AWS keys come from?


Git
app code
Git
debian pkg
aws
provisioning
code
launch
aws
apply
puppet
func
test
terminate
aws
deploy
prod
prod db
node
prod web
node
puppet
Git
tgz

create stage to push to production

Use your fresh git working copies as the starting point.

go pseudo-code

- deploy stage
  - deploy_db job
    - fetch artifact puppet.tgz
    - ./deploy_to_db.sh
  - deploy_web job
    - fetch artifacts puppet.tgz opencart.db
    - ./deploy_to_web.sh

```
scp ... opencart.deb puppet.tgz vagrant@db:

ssh ... vagrant@... “tar xvf puppet.tgz”

ssh ... vagrant@... “sudo puppet apply --modulepath=modules db.pp”
```

Command: bash
Arguments: -c ssh -o StrictHostKeyChecking=no -i /var/go/infra_lab.pem ubuntu@web.part2.com sudo FACTER_database_password=$FACTER_database_password FACTER_database_host=$FACTER_database_host puppet apply --modulepath=modules web.pp

Applying this workshop to your projects

How could Infrastructure as Code help on your projects?

What concerns are likely to arise from ITOps?

How can you overcome resistance?

How could monitoring help on your projects?

Q&A

Thank you

(don’t forget to clean your /etc/hosts file! :-)