



Grant Allen
Michael Still
Google Inc.
April, 2007

Managing MySQL the Slack Way

How Google Deploys New MySQL Servers

Grant Allen
Michael Still
Google, Inc.
April, 2007

Agenda

We thought we should talk about these things:

- *The importance of repeatable software installs*
- *How we configure software using Slack*
- *How we generate MySQL configuration files*
- *How we bootstrap new MySQL servers*

Repeatable software installs

It's important to be able to deploy effectively identical MySQL servers quickly

- Hardware failure
- Handling more load
- Disaster recovery

Repeatable software installs

Building an entire machine has several steps

- Installing the operating system
- Installing the software
- Configuring the software
- Bootstrapping the initial data

Repeatable software installs

Building an entire machine has several steps

- Installing the operating system
- Installing the software
- Configuring the software
- Bootstrapping the initial data

We're going to focus on the last three in this talk

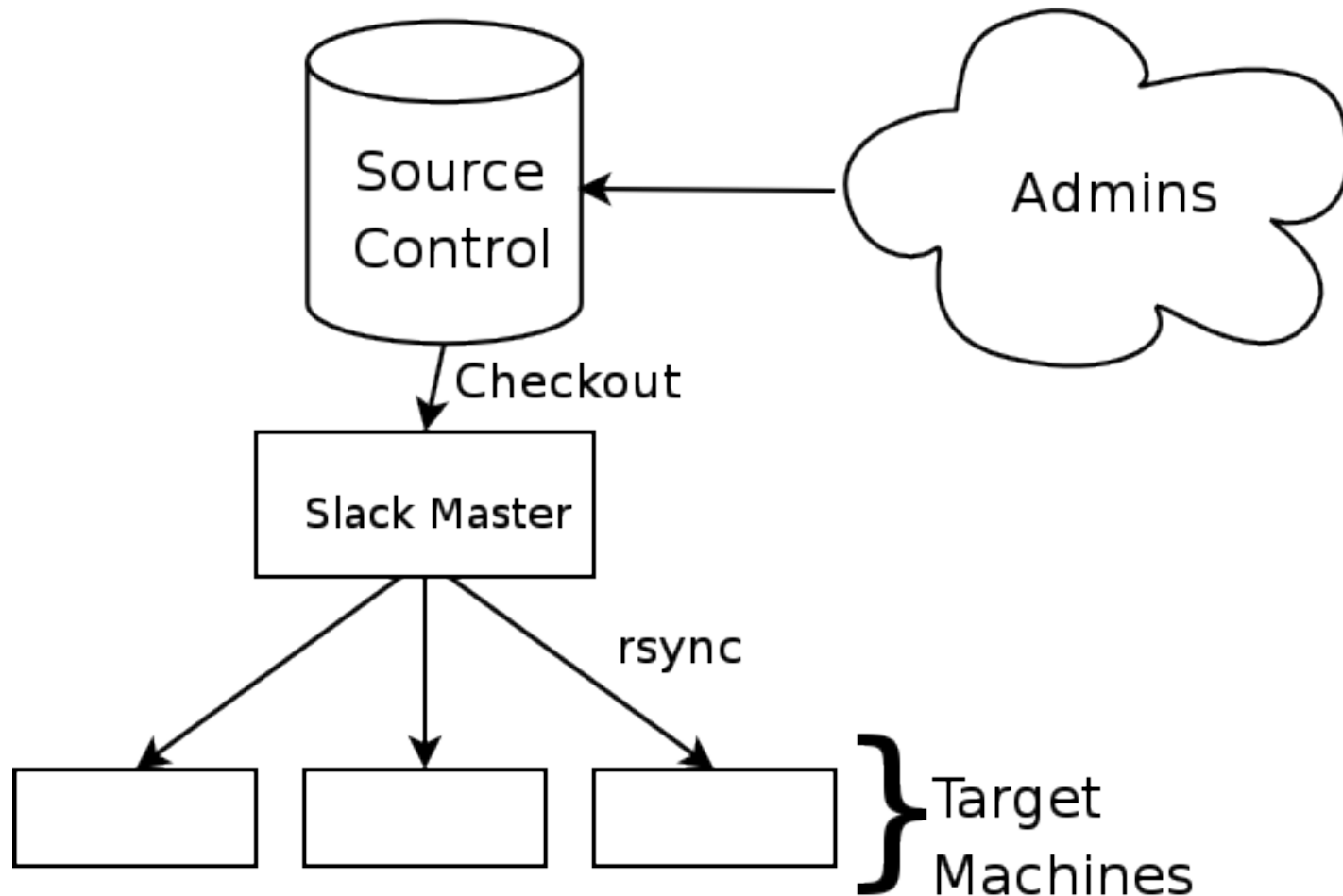
Repeatable software installs

slack

- Google's home grown software deployment system
- Written by Alan Sundell and Roman Marxer

- Centralized configuration repository
- Configuration is then deployed onto selected machines
- You can have more than one “slack role” on a given machine

Repeatable software installs



Repeatable software installs

slack

- Configuration changes flow from the Administrators via a code review mechanism
- Once checked into the source control repository, they are synced to the slack master
- Individual servers then sync their configurations from the slack master
 - Manually: *slack role.sub-role*
 - Automatically via cron

Repeatable software installs

slack

- What does a role look like?

```
$ find mail-loghost -type f
```

```
mail-loghost/scripts/preinstall
```

```
mail-loghost/files/etc/cron.d/logcleanup
```

```
mail-loghost/files/etc/syslog-  
ng.conf.mailloghost
```

```
mail-loghost/files/usr/local/scripts/logcleanup
```

```
mail-loghost/scripts/postinstall
```

```
$
```

Repeatable software installs

slack

- What does a sub-role look like?

```
$ find mail-loghost -type f
```

```
mail-loghost/scripts/preinstall
```

```
mail-loghost/files/etc/cron.d/logcleanup
```

```
mail-loghost/files/etc/syslog-  
ng.conf.mailloghost
```

```
mail-loghost/files/usr/local/scripts/logcleanup
```

```
mail-loghost/files.trakken-mail/etc/syslog-  
ng.conf.mailloghost
```

```
mail-loghost/scripts/postinstall
```

Repeatable software installs

slack

- There are lot of similarities between this and binary packages like .deb or RPM:
 - Pre and post install scripts
 - Deploys files
- Things which are cool about slack:
 - There is no intermediate packaged form
 - Sub-roles
 - Pre and post install scripts in any language

Repeatable software installs

slack

- Versioning
 - How do I undeploy a slack role?
 - How do I find out what slack roles a machine needs?
 - Dependencies between slack roles?

Repeatable software installs

slack

- Where can you get slack?
 - Google and Alan Sundell have released slack!
 - <http://www.sundell.net/~alan/projects/slack/>

Generating the MySQL configuration

MySQL configuration files differ per machine:

- server-id
- Replication configuration
- We use simple text substitutions and some scripting in the Slack post install script to generate this file

```
# The server ID comes from substitution based on IP address  
server-id=SERVERID
```

```
# The replication configuration comes from a simple shell  
# script  
REPLICATION
```

Bootstrapping the database

Slow and steady

- Goal:
 - Get a new copy of the database in place ASAP
 - Without (total) service interruption
 - In a fully automated fashion

Bootstrapping the database

Slow and steady

- Goal:
 - Get a new copy of the database in place ASAP
 - Without (total) service interruption
 - In a fully automated fashion
- Using:
 - Restoring a backup? ... slow, but will eventually get there

Bootstrapping the database

Slow and steady

- Goal:
 - Get a new copy of the database in place ASAP
 - Without (total) service interruption
 - In a fully automated fashion
- Using:
 - Restoring a backup? ... slow, but will eventually get there
 - Cloning an existing replica (without too much service impact)

Bootstrapping the database

Fast

- Cloning the database requires copying the constituent files:
 - Lots of tools available ... everything from cp to rsync, NFS to SMB
 - File copying needs to be *fast*
 - High-end (and expensive) technology such as SAN splitting, etc.

- But:
 - Other dependencies increase complexity and risk
 - Totally secure environment removes some options
 - Fault-tolerance would be nice

Bootstrapping the database

Faster

- scp the files from a suspended replica:
 - The Porsche Boxter approach
 - Avoids the statement replay drag from regular restores
- But:
 - You need a replica that can be suspended temporarily
 - scp not the fastest of mechanisms

Bootstrapping the database

Fastest!

- nc, tar, and automagical scripting
 - The McLaren Mercedes F1 approach
 - Avoids the statement replay drag from regular restores
 - Squeeze every last bit of omph from your environment

- But:
 - Again, you need a replica that can be suspended temporarily

Bootstrapping the database

Fastest!

- On the target:
 - Start nc listening for incoming data

```
nc -l -p 12345 | tar xvf -
```

- On the source:
 - Push tar'ed files to the target

```
tar cvf - * | nc target 12345
```

Bootstrapping the database

Fastest!

- Sounds easy, but gets complicated by:
 - Terabytes of data
 - Slow WAN links
- Add scripting magic to run copies in parallel and tolerate failures:

```
nc -l -p port_for_file | tar xvf -  
tar cvf - specific_file | nc target port_for_file
```

Bootstrapping the database

Cleaning up ready for launch

- Cloned replica has personality issues:
 - Relay bin log using source host's name
 - Master not currently aware of new replica
('replica_user'@% not exactly secure)
 - Possible left-over files from old host (depending on copy method)

Bootstrapping the database

Cleaning up ready for launch

- Easily Solved:
 - Script to identify new replica with master
 - Script to start and stop mysqld to generate bin log stub index file

```
cat new.host.name-relay-bin.index >>
```

```
old.host.name-relay-bin.index
```

```
mv old.host.name-relay-bin.index
```

```
new.host.name-relay-bin.index
```

- Script to clean up any orphaned files from clone source

Bootstrapping the database

More complex scenarios

- Daisy Chain / Cascade replicas
 - By varying the slack subrole, we can activate the bin log and `log_slave_updates` for the intermediaries
 - bin log relay cleanup includes one-time options to either start or skip slave startup, depending on desired start-up characteristics

Questions?

Any questions?



Grant Allen
Database Administrator
Google, Inc.
fuzz@google.com

Michael Still
Site Reliability Engineer
Google, Inc.
michaelstill@google.com

